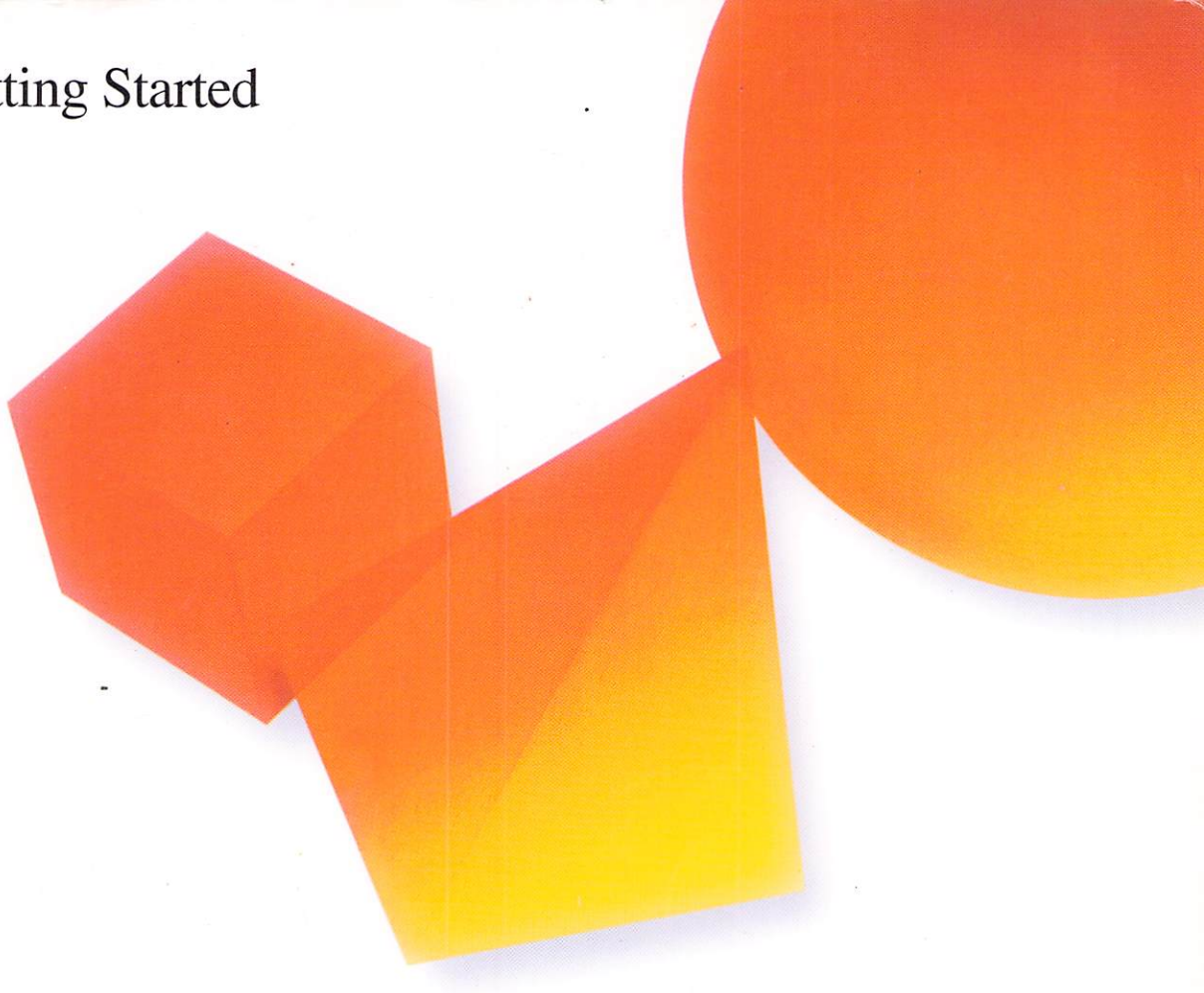
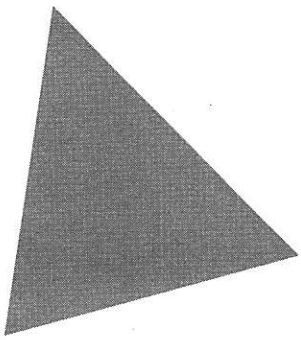
Getting Started



Borland®
InterBase® Server

for Windows 95 & Windows NT

Getting Started



Borland®
InterBase®
for Windows 95 and Windows NT

Borland International, Inc., 100 Borland Way
P.O. Box 660001, Scotts Valley, CA 95067-0001

Borland may have patents and/or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.

Copyright © 1992, 1995 Borland International. All rights reserved. All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

INT1340WW21774 1EOR1295
9596979899-9 8 7 6 5 4 3 2 1
H1

Table of Contents

Preface 1

The InterBase Documentation 1
Printing Conventions 2
Text Conventions 2
Syntax Conventions 3
Database Object-naming Conventions . . . 4
File-naming Conventions 4
Primary File Specifications 5
Secondary File Specifications 5

Part 1: Overview

Chapter 1: Introduction 9

What Is InterBase on Windows 95 and Windows NT? 9
Local InterBase, Single-user 9
InterBase Server, Multi-User 9
InterBase Features 12
SQL Support 13
Interactive SQL 14
Database Access 14
Transaction Management 14
Multi-generational Architecture 15
Database Administration 15
Managing Server Security 15
Performing Database Backup and Recovery 16
Maintaining a Database 16
Viewing Statistics 17
Where to Find Information 17

Chapter 2: Using InterBase 19

Client-server Development 19
Database/Application Development Methodology 20
Design Phase 22
Implementation Phase 22
Deployment Phase 23
Building Databases 24
Tables 25
Columns 25
Data Types 25
Numeric Data Types 26
Character Data Types 26
Date Data Types 26
BLOB Data Types 26
Arrays of Data Types 27
Domains 27
Referential Integrity Constraints 27
Indexes 27
Views 28
Stored Procedures 28
Triggers 29
Event Alerters 29
User-defined Functions 30
BLOB Filters 30
Application Development 31
Borland Client Tools Applications . . . 31
Embedded Applications 31
Predefined Database Queries 32
Dynamic Applications 32
API Applications 32
Multi-database Applications 33
Communications Protocols 33

Part 2: Data Definition Tools

Chapter 3: Using Windows ISQL 37

Starting and Exiting Windows ISQL . . . 37
The ISQL Window 38
Getting Help 38
Exiting Windows ISQL 39
Temporary Files 39
Connecting to an Existing Database . . . 39
Creating a New Database 40
Dropping a Database 42
Disconnecting from a Database 42
Executing SQL Statements 42
Executing SQL Interactively 42

Legal Statements 43
Executing an ISQL Script File. 44
Committing and Rolling Back Work. 44
Saving Results to a File 45
Saving ISQL Output 45
Saving the Session 46
Extracting Metadata 46
Extracting Database Metadata 46
Extracting Table Metadata 48
Extracting View Metadata 48
Changing Windows ISQL Settings 49
Basic ISQL Settings 49
Advanced ISQL Settings 50
BLOB Display 51
Character Set 51
Displaying Settings 51
Displaying Version Information 52
Displaying Database Information and Metadata. 52

Chapter 4: Using ISQL Script Files 55

Creating and Executing SQL Files. 55
Creating an ISQL Script File 55
Executing an ISQL Script File. 56
Committing and Rolling Back Work. 56
Adding Comments 57
ISQL SET Statements 57

Part 3: Managing the InterBase Server

Chapter 5: Starting and Stopping the Server 73

Starting InterBase on Windows 95 73
Selecting Server Startup Options. . . . 73
Shutting Down InterBase on Windows 95 74
Starting InterBase on Windows NT 74
Selecting Server Startup Options. . . . 74
Shutting Down the Server on Windows NT 76
Viewing Server Properties 76

Chapter 6: Introduction to Server Manager 77

The Server Manager Window 78
Server Manager Menus 79
Toolbar. 79
Server/Database Tree Pane 80
Summary Information Area 82
Standard Text Display Window. 82

Chapter 7: Server Logon and Database Access. 83

Logging in to a Server 83
Connecting to a Database 84
Examples 85
Administering Security 85
The SYSDBA User 86
Adding a User. 86
Modifying User Configurations. 87
Deleting a User 87

Chapter 8: Maintaining a Database 89

Selecting a Database 89
Shutting Down a Database 90
Denying New Connections 91
Denying New Transactions 92
Forcing Shutdown 92
Restarting a Database 93
Viewing Active Database Connections. 93
Configuring Database Properties 94
Adjusting Database Sweeping 94
Overview of Sweeping 95
Setting the Sweep Interval 96
Disabling Automatic Sweeping 96
Controlling Performance of Forced Writes. 96

Two-phase Commit and Transaction Recovery 97
Recovering Transactions 98
Performing an Immediate Database Sweep 100
Validating and Repairing a Database 100
Validation Options. 101
Handling Checksum Errors 102
Repairing a Corrupt Database 102

Chapter 9: Backing Up and Restoring a Database. 105

Using the Backup and Restore Utilities 105
Backing Up a Database 106
Backup Options 107
Transportable Format 107
Back Up Metadata Only 108
Disable Garbage Collection. 108
Ignore Transactions in Limbo . . . 108
Ignore Checksums. 109
Verbose Output 109
Restoring a Database 110
Restoring to Multiple Files. 111
Restore Options 111
Start Page. 112
Page Size 112
Replace Existing Database 112
Commit after Each Table 113
Deactivate Indexes 113
Do Not Restore Validity Conditions 113
Verbose Output 114
Upgrading to a New On-disk Structure 114

Index 117

Tables and Figures

Table P-1: InterBase Documentation 1
Table P-2: Text Conventions. 2
Table P-3: Syntax Conventions 3
Figure 1-1: InterBase Client/Server Connections 11
Table 1-1: InterBase Features 12
Table 1-2: InterBase Information Roadmap. 17
Figure 2-1: Database and Application Development Lifecycle 21
Figure 2-1: Database Objects 24
Table 3-1: Order of Metadata Extraction 47
Table 3-2: Basic ISQL Settings 49
Table 3-3: Metadata Information Items 52
Table 4-1: SET Statements 57
Table 6-1: Server/Database Tree Commands 81
Table 6-2: Pop-up Menu from Server 81
Table 6-3: Pop-up Menu from Database. 81

Preface

This preface describes the documentation, the printing conventions used to display information in text and in code examples, and the conventions a user should employ when specifying database objects and files by name in applications.

The InterBase Documentation

The InterBase Server documentation set is an integrated package designed for all levels of users of the InterBase Server or Local InterBase.

Table P-1: InterBase Documentation

Book	Description
Getting Started, InterBase for Windows 95 and Windows NT	Introduces the user to both Local InterBase and InterBase Server that run on Windows 95 and Windows NT. Provides information on how to use Windows ISQL and Server Manager.
Installing and Running on . . .	Platform-specific information on installing and running InterBase on Unix and NetWare.
Data Definition Guide	Explains how to create, alter, and delete database objects, such as tables, indexes, UDFs, stored procedures, and triggers.
Language Reference	Describes SQL and DSQL syntax and usage.
Programmer's Guide	Describes how to write embedded SQL and DSQL database applications in a host language, precompiled through **gpre**. (InterBase Server Developer Edition)
API Guide	Explains how to write database applications using the InterBase API. (InterBase Server Developer Edition)

Table P-1: InterBase Documentation

Book	Description
Online Help	Comprehensive online Help is provided with InterBase running on Windows platforms, including: • information on how to define and maintain an InterBase database • a tutorial for users who are not familiar with defining a database or building queries to select data • information on the InterBase online tools, Windows ISQL and Server Manager (information on command line tools is also provided) • information on remote configuration • a full reference describing the InterBase implementation of SQL • information on programming InterBase custom applications (InterBase Server Developer Edition) • information on the InterBase API, **gpre** preprocessor, and developing embedded systems (InterBase Server Developer Edition)

Printing Conventions

The InterBase documentation set uses different fonts to distinguish various kinds of text and syntax.

Text Conventions

The following table describes font conventions used in text, and provides examples of their use:

Table P-2: Text Conventions

Convention	Purpose	Example
UPPERCASE	SQL keywords, names of all database objects such as tables, columns, indexes, stored procedures, and SQL functions.	The following SELECT statement retrieves data from the CITY column in the CITIES table.
italic	Introduces new terms, and emphasizes words. Also used for file names and host-language variables.	The *ISC4.GDB* security database is *not* accessible without a valid *username* and *password*.

Table P-2: Text Conventions

Convention	Purpose	Example
bold	Utility names, user-defined and host-language function names. Function names are always followed by parentheses to distinguish them from utility names.	To back up and restore a database, use **gbak** or the server manager. The **datediff()** function can be used to calculate the number of days between two dates.

Syntax Conventions

The following table describes the conventions used in syntax statements and sample code, and offers examples of their use:

Table P-3: Syntax Conventions

Convention	Purpose	Example
UPPERCASE	Keywords that must be typed exactly as they appear when used.	SET TERM !!;
italic	Parameters that *cannot* be broken into smaller units. For example, a table name cannot be subdivided.	CREATE TABLE *name* (*<col>* [, *<col>* ...]);
<italic>	Parameters in angle brackets that *can* be broken into smaller syntactic units.	CREATE TABLE *name* (*<col>* [, *<col>* ...]);
	For example, column definitions (*<col>*) can be subdivided into a name, data type and constraint definition.	*<col>* = *name <datatype>* [CONSTRAINT *name <type>*]
[ ]	Square brackets enclose optional syntax.	*<col>* [, *<col>* ...]
...	Closely spaced ellipses indicate that a clause within brackets can be repeated as many times as necessary.	(*<col>* [, *<col>* ...]);

Table P-3: Syntax Conventions (Continued)

Convention	Purpose	Example
|	The pipe symbol indicates that either of two syntax clauses that it separates may be used, but not both. Inside curly braces, the pipe symbol separates multiple choices, one of which *must* be used.	SET TRANSACTION {SNAPSHOT [TABLE STABILITY] | READ COMMITTED};
{ }	Curly braces indicate that one of the enclosed options *must* be included in actual statement use.	SET TRANSACTION {SNAPSHOT [TABLE STABILITY] | READ COMMITTED};

Database Object-naming Conventions

InterBase database objects, such as tables, views, and column names, appear in text and code in uppercase in the InterBase documentation set because this is the way such information is stored in a database's system tables.

When an applications programmer or end user creates a database object or refers to it by name, case is unimportant. The following limitations on naming database objects must be observed:

- Start each name with an alphabetic character (A-Z or a-z).
- Restrict object names to 31 characters, including dollar signs ($), underscores (_), 0 to 9, A to Z, and a to z. Some objects, such as constraint names, are restricted to 27 bytes in length.
- Keep object names unique. In all cases, objects of the same type, for example, tables and views, *must* be unique. In most cases, object names must also be unique within the database.

For more information about naming database objects with CREATE or DECLARE statements, see the *Language Reference*.

File-naming Conventions

InterBase is available on a wide variety of platforms. In most cases users in a heterogenous networking environment can access their InterBase database files

regardless of platform differences between client and server machines if they know the target platform's file naming conventions.

Because file-naming conventions differ widely from platform to platform, and because the core InterBase documentation set is the same for each of these platforms, all file names in text and in examples are restricted to a base name with a maximum of eight characters, with a maximum extension length of three characters. For example, the example database on all servers is referred to as *employee.gdb*.

Generally, InterBase fully supports each platform's file-naming conventions, including the use of node and path names. InterBase, however, recognizes two categories of file specification in commands and statements that accept more than one file name. The first file specification is called the *primary file specification*. Subsequent file specifications are called *secondary file specifications*. Some commands and statements place restrictions on using node names with secondary file specifications.

In syntax, file specification is denoted as follows:

```
"<filespec>"
```

Primary File Specifications

InterBase syntax always supports a full file specification, including optional node name and full path, for primary file specifications. For example, the syntax notation for CREATE DATABASE appears as follows:

```
CREATE {DATABASE | SCHEMA} "<filespec>"
   [USER "username" [PASSWORD "password"]]
   [PAGE_SIZE [=] int]
   [LENGTH [=] int [PAGE[S]]]
   [DEFAULT CHARACTER SET charset]
   . . .
```

In this syntax, the *<filespec>* that follows CREATE DATABASE supports a node name and path specification, including a platform-specific drive or volume specification.

Secondary File Specifications

For InterBase syntax that supports multiple file specification, such as CREATE DATABASE, all file specifications after the first are secondary. Secondary file specifications generally cannot include a node name, but may specify a full path name. For example, the syntax notation for CREATE DATABASE appears as follows:

```
CREATE {DATABASE | SCHEMA} "<filespec>"
   [USER "username" [PASSWORD "password"]]
   [PAGE_SIZE [=] int]
   [LENGTH [=] int [PAGE[S]]]
   [DEFAULT CHARACTER SET charset]
   [<secondary_file>]

<secondary_file> = FILE "<filespec>" [<fileinfo>] [<secondary_file>]

<fileinfo> = LENGTH [=] int [PAGE[S]] | STARTING [AT [PAGE]] int
   [<fileinfo>]
```

In the secondary file specification, *<filespec>* does not support specification of a node name.

PART 1
Overview

Part 1 provides an introduction to the InterBase Server and describes its features.

Chapter 1: "Introduction" gives a general overview of InterBase and introduces each of the features.

Chapter 2: "Using InterBase" provides an overview of database application development.

CHAPTER 1

Introduction

This chapter provides a high-level introduction to the Borland InterBase Server for Windows 95 and Windows NT.

What Is InterBase on Windows 95 and Windows NT?

InterBase is a relational database management system (RDBMS) that provides rapid transaction processing and data sharing in a single- or multi-user environment. At its core, InterBase is a server technology that offers transparent support across heterogenous networks. InterBase runs on Windows 95, Windows NT, Novell NetWare, and many implementations of the Unix operating systems.

InterBase running on Windows 95 and Windows NT is provided in two forms: Local InterBase (single-user) and InterBase Server (remote, multi-user). Both servers come with Windows client tools: Windows ISQL for data definition and manipulation and Server Manger for user authorization and database administration.

Local InterBase, Single-user

As a single-user, 32-bit ANSI SQL-compliant local server, InterBase ships with client/server versions of Borland's client development tools, such as Delphi and Visual dBASE. The single-user version resides on the developer's PC and is commonly used as a local database environment to develop stand-alone desktop SQL applications and client/server applications with remote server connections.

InterBase Server, Multi-User

As a multi-user, 32-bit ANSI SQL-compliant remote server, InterBase is sold separately or shipped with a limited number of user licenses in client/server versions of Borland's client development tools such as Delphi. InterBase Server is

the full-featured enterprise server that resides on hardware separate from the application.

Developers can build client/server applications using Borland's client development tools that use the Borland Database Engine (BDE), Borland's core database and connectivity middleware. Access to InterBase databases is provided through the InterBase SQL Link.

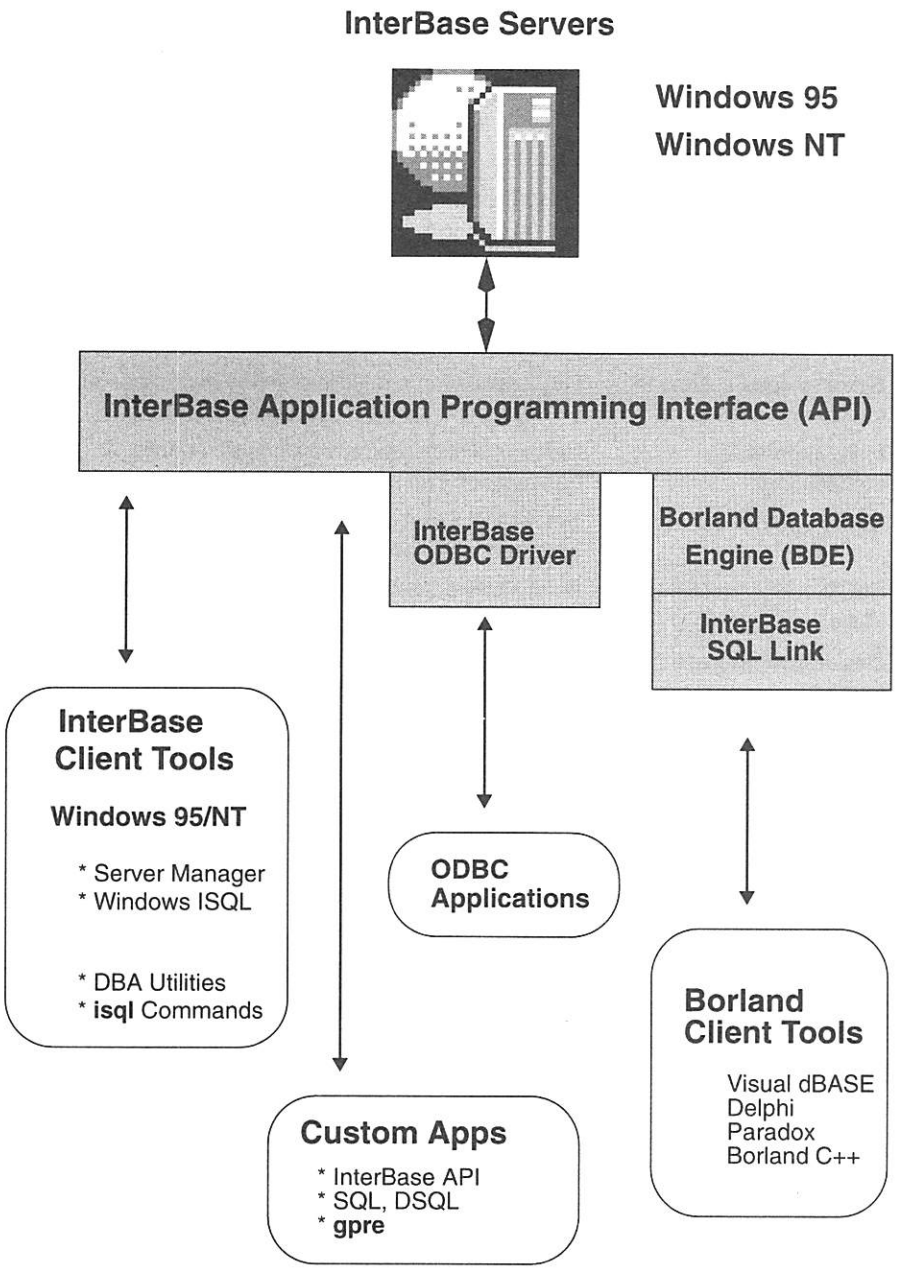
When included with Borland client tools, such as Delphi Client/Server, the BDE acts as the API interface to the InterBase Server. Applications that interact directly with the InterBase API must be created using the InterBase Server Developer Edition.

With InterBase Server, custom client applications can also be written using SQL and dynamic SQL (DSQL) statements embedded within a host language. In addition, these applications can be written with low-level InterBase API functions. InterBase access is provided directly through the InterBase API.

In addition, InterBase Server includes a driver for the Open Database Connectivity standard (ODBC) that enables ODBC client applications to share data with InterBase servers.

Figure 1-1 shows the relationship of Local InterBase and InterBase Server to client applications.

Figure 1-1: InterBase Client/Server Connections



InterBase Features

InterBase on Windows 95 and Windows NT offers all the benefits of a fully relational RDBMS. The following table lists some of the key InterBase features:

Table 1-1: InterBase Features

Feature	Description
Network protocol support	Support for both Microsoft NetBEUI/Named Pipes and TCP/IP for communication to clients. InterBase Server only.
SQL-92 entry-level conformance	ANSI standard SQL, available through an Interactive SQL tool and Borland desktop applications.
Simultaneous access to multiple databases	One application can access many databases at the same time.
Multi-generational architecture	Server maintains older versions of records (as needed) so that transactions can see a consistent view of data.
Query optimization	Server optimizes queries automatically, or user may manually specify query plan.
BLOB data type and BLOB filters	Binary Large Objects can contain unformatted data such as graphics and text.
Declarative referential integrity	Automatic enforcement of cross-table relationships (between FOREIGN and PRIMARY KEYs.)
Stored procedures	Programmatic elements in the database for advanced queries and data manipulation actions.
Triggers	Self-contained program modules that are activated when data in a specific table is inserted, updated, or deleted.
Event alerters	Messages passed from the database to an application. Enables applications to receive asynchronous notification of database changes.
Updatable views	Views can reflect data changes as they occur.
User-defined functions (UDFs)	Program modules that run on the server.
Outer joins	Relational construct between two tables that enables complex operations.
Explicit transaction management	Full control of transaction start, commit, and rollback, including named transactions.
Concurrent multiple application access to data	One application reading a table does not necessarily block others from it.
Multi-dimensional arrays	Column data types arranged in an indexed list of elements.

Table 1-1: InterBase Features (Continued)

Feature	Description
Automatic two-phase commit	Multi-database transactions check that changes to all databases happen before committing. (InterBase Server only)
InterBase API	Low-level functions that enable applications to construct SQL/DSQL statements directly to the InterBase engine and receive results back. InterBase Server Developer Edition.
gpre	Preprocessor for converting embedded SQL/DSQL statements and variables into a format that can be read by a host-language compiler. InterBase Server Developer Edition.
Server Manager	Windows tool for database backup, restoration, maintenance, and security.
Windows ISQL	Interactive data definition and query tool for Windows
Command-line ISQL	Command-line version of the InterBase Interactive SQL Tool. Can be used instead of Windows ISQL.
Command-line DBA utilities	Command-line version of the InterBase database administration tools. Can be used instead of Server Manager.
Header files	Files included at the beginning of application programs that define InterBase data types and function calls. InterBase Server Developer Edition.
Example make files	Files that demonstrate how to invoke the Borland MAKE utility to compile and link InterBase applications under Windows. InterBase Server Developer Edition.
Example programs	C programs, ready to compile and link, which may be used to query standard InterBase example databases on the server. InterBase Server Developer Edition.
Message file	INTERBAS.MSG, containing messages presented to the user.

SQL Support

InterBase conforms to entry-level SQL-92 requirements. It supports declarative referential integrity, updatable views, and outer joins. InterBase Server provides libraries that support development of embedded SQL and DSQL client applications. On all InterBase Server platforms, client applications can be written to the InterBase API, a library of low-level engine calls.

InterBase also supports extended SQL features, some of which anticipate SQL3 extensions to the SQL standard. These include stored procedures, triggers, and segmented BLOB support.

For information on SQL, see the online Help or the *Language Reference*.

Interactive SQL

With Local InterBase and InterBase Server, you can issue interactive SQL commands with Windows ISQL, a Windows application running on any client PC connected to Local InterBase or InterBase Server. For more information about using Windows ISQL, see the online Help or Chapter 3: "Using Windows ISQL." With InterBase Server, you can use command-line ISQL. For information on using command-line ISQL, see the online Help.

Note The Interactive SQL utility is referred to in this manual as ISQL. In the *Language Reference* and *Data Definition Guide*, it is referred to as **isql**, since these books also support platforms on which commands are case-sensitive. On Windows NT and Windows 95, the two are interchangeable.

Database Access

InterBase enables client applications to access a single database or to access multiple databases simultaneously. Separate client applications can also access the same databases simultaneously. To control database access and ensure data integrity, InterBase provides automatic two-phase commit. SQL triggers can notify client applications when specific database events occur, such as insertions or deletions.

User-defined functions (UDFs) can be written and stored in an InterBase database, where they are accessible to all client applications accessing the database.

Two-phase commit is supported only on Windows NT and Windows 95 InterBase Servers.

Transaction Management

Client applications can start multiple simultaneous transactions. InterBase provides full and explicit transaction control for starting, committing, and rolling back transactions. The statements and functions that control starting a transaction also control transaction behavior.

InterBase transactions can be isolated from changes made by other concurrent transactions. For the life of these transactions, the database will appear to be unchanged except for the changes made by the transaction. Records deleted by another transaction will exist, newly stored records will not appear to exist, and updated records will remain in the original state.

For information on transaction management, see the online Help or the *Programmer's Guide*.

Multi-generational Architecture

InterBase provides expedient handling of time-critical transactions through support of data concurrency and consistency in mixed use (query and update) environments. InterBase uses a *multi-generational architecture*, which creates and stores multiple versions of each data record. By creating a new version of a record, InterBase allows all users to read a version of any record at any time, even if another user is changing that record. InterBase also uses transactions to isolate groups of database changes from other changes.

Database Administration

InterBase provides both Windows-based and command-line tools for managing databases and servers.

You can perform database administration (DBA) on databases residing on Local InterBase or InterBase Server with Server Manager, a Windows application running on a client PC. For more information about using the Server Manager, see the online Help or Chapter 6: "Introduction to Server Manager."

You can also use command-line DBA utilities on the server. For information on using the command-line DBA utilities, see the online Help.

Server Manager and the command-line utilities enable the DBA to:

- Manage server security.
- Back up and restore a database.
- Perform database maintenance.
- View database and lock manager statistics.

Managing Server Security

InterBase maintains a list of user names and passwords in a security database. The security database allows clients to connect to an InterBase database on a server if a user name and password supplied by the client match a valid user name and password combination in the security database on the server.

You can add and delete user names and modify a user's parameters, such as password and user ID.

For information about managing server security, see the online Help or Chapter 7: "Server Logon and Database Access."

Performing Database Backup and Recovery

You can back up and restore a database using Server Manager or the command-line DBA utilities. A backup can run concurrently with other processes accessing the database because it does not require exclusive access to the database.

Database backup and restoration can also be used for:

- Erasing obsolete versions of database records
- Changing the database page size
- Changing the database from single-file to multi-file
- Transferring a database from one operating system to another

Server Manager and the command-line backup tool also have an option for backing up only a database's metadata to recreate an empty database.

For information about database backup and recovery, see the online Help or Chapter 9: "Backing Up and Restoring a Database."

Maintaining a Database

You can prepare a database for shutdown and perform database maintenance using Server Manager or command-line DBA utilities. If a database incurs minor problems, such as an operating system write error, these tools enable you to sweep a database without taking the database off-line.

Some of the tasks that are part of database maintenance are:

- Sweeping a database
- Shutting down the database to provide exclusive access to it
- Validating table fragments
- Preparing a corrupt database for backup
- Resolving transactions "in limbo" from a two-phase commit
- Validating and repairing the database structure

For information on database maintenance, see the online Help or Chapter 8: "Maintaining a Database."

Viewing Statistics

Server Manager enables the DBA to monitor the status of a database by viewing statistics from the database header page, and an analysis of tables and indexes.

Where to Find Information

The InterBase documentation is an integrated system designed for all levels of users of InterBase Server and Local Interbase. Information about using InterBase is provided in three forms: print documentation, online Help (superset), and Borland OnLine (http://www.borland.com),

Table 1-2: InterBase Information Roadmap

Information about...	Source
Installing InterBase	Online Help during install (PC platforms) Print: *Installing and Running on...* (Unix) WWW: Borland OnLine (Unix)
Remote configuration	Online Help (PC platforms)
Conceptual overview (white papers)	Borland OnLine
Supported features	Online Help (PC platforms) Print: *Getting Started...*
Learning to use SQL	Online Help tutorial (PC platforms)
Database Design	Online Help (PC platforms) Print: *Data Definition Guide*
SQL Syntax	Online Help (PC platforms) Print: *Language Reference*
Windows ISQL	Online Help (PC platforms) Print: *Getting Started...*
Command-Line ISQL	Online Help (PC platforms) Print: *Installing and Running on... (Unix)*
Server Manager	Online Help (PC platforms) Print: *Getting Started....*
Command-Line DBA utilities	Online Help (PC platforms) Print: *Installing and Running on...(Unix)*
Application development overview	Print: *Getting Started...*
Embedded SQL/DSQL applications (Developer Edition)	Online Help (PC platforms) Print: *Programmer's Guide*

Table 1-2: InterBase Information Roadmap

Information about...	**Source**
API applications (Developer Edition)	Online Help (PC platforms) Print: *API Guide*
Supported compilers	online readme (PC platforms) Borland OnLine (Unix)
Example database	/examples (described in online Help)
Example programs (developer Edition)	/examples
Borland client development tools	Print and online Help for Borland products, such as Delphi, Visual dBASE, Borland C++, Paradox

CHAPTER 2

Using InterBase

This chapter describes client-server database/application development methodology and introduces some of the concepts important in building databases and applications with InterBase.

Client-server Development

Developing client/server databases and applications is similar in many ways to developing other types of software, but there are some important distinctions and challenges that must be addressed. The methodology presented in this section should be used as a guideline that can be adapted to meet your specific business needs.

A database by itself is rarely useful; an application usually provides the mechanism and interface for end users to access it. An InterBase client-server database application consists of server elements and client elements. Because InterBase itself is a server, most of the documentation is concerned with the database and server aspects. Often, the server/database and client/application elements are intertwined, so to provide context, this discussion considers both.

Before discussing the development process, it is important to define some terminology. The terms "client" and "server" are used conceptually rather than physically. In practice, the client is usually a desktop PC running Windows and the server is a workstation or high-performance platform running a multi-user operating system such as Unix or Windows NT. However, this is not necessarily the case. InterBase allows a platform to be both client and server. In most cases, it will be more cost-efficient to run client elements on inexpensive desktop PCs, and run server elements on more expensive machines.

Server elements include the database engine, the database data, and ancillary processes that execute on the server such as user-defined functions (UDFs), BLOB filters, triggers, and stored procedures. All of these server elements are part of the database itself or closely related to it, and run on the server platform.

Client elements run on the client, and include embedded SQL and DSQL or InterBase API applications that provide the user interface (with related GUI presentation system, such as Windows), middleware such as the Borland Database Engine (BDE) or ODBC driver, and low-level network drivers for communication protocols such as TCP/IP. Other client elements include InterBase tools such as Windows ISQL (for data definition) and Server Manager (for database administration), and application development tools such as Delphi, Borland ReportSmith, dBASE, and Paradox.

The term "database/application development" refers both to building the database and associated server elements and building the application and associated client elements.

Database/Application Development Methodology

The goal of client-server database/application development is to build a product which meets end users' long-term needs. While this goal may seem obvious, it is important not to lose sight of it throughout the complexities and often conflicting demands of the development process. To create a successful application, it is critical to define the end users' needs clearly and in detail early in the development process.

The three primary stages of database/application development are:

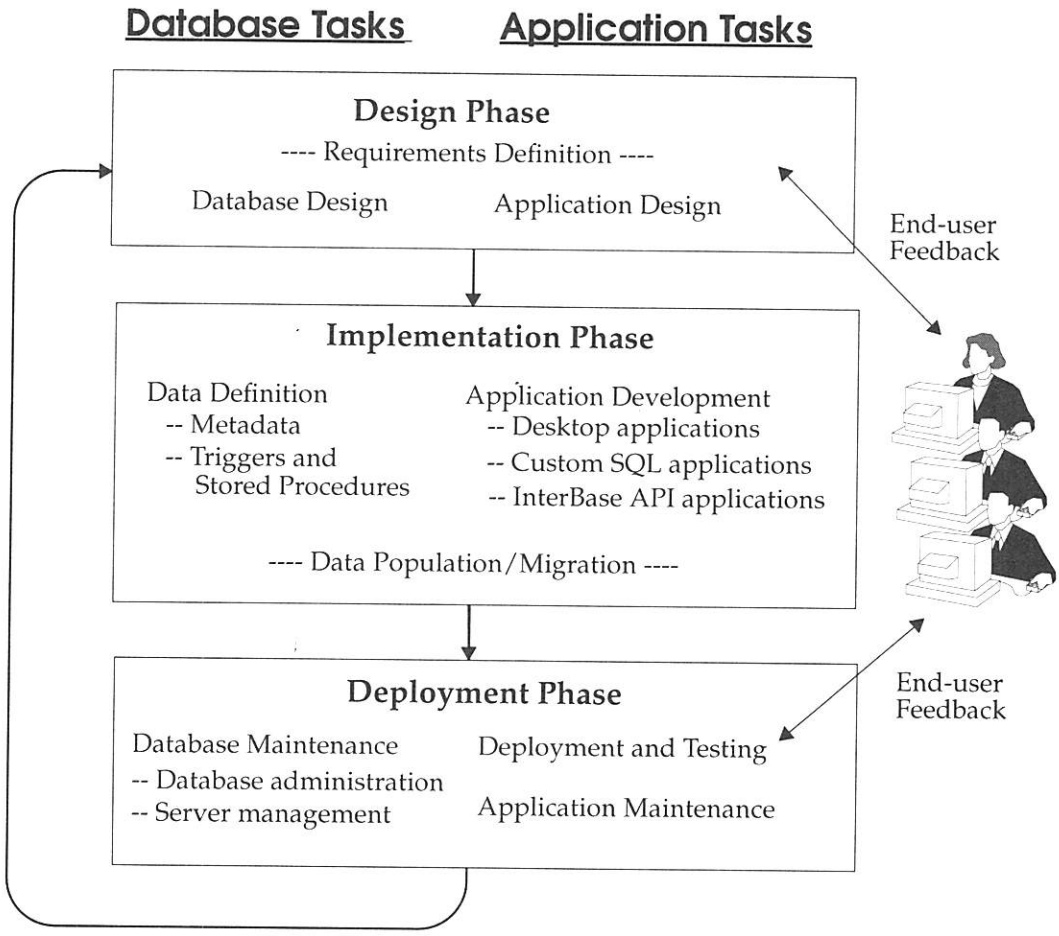
- Design
- Implementation
- Deployment and maintenance

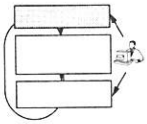
As illustrated in Figure 2-1, there are database and application tasks in each of these phases. Depending on the size and scope of the development project, the database and application tasks may be performed by different individuals or by the same individual. Often, one team or individual will be responsible for the database tasks of the project, and another team or individual will be responsible for the application tasks.

When development responsibilities are thus divided it is important to clearly delineate in the design phase which functions will be performed by the database (typically, on the server) and which will be performed by the application (typically, on the client). Usually, the functional lines are clear cut. But database processes such as UDFs and stored procedures can sometimes perform functions that can also be performed by the client application. Depending on the expected deployment configuration, the design can allocate such functions to either client or server.

It is also important to realize that database/application development is by its nature an iterative process. Users may not fully understand their own needs, or may define additional needs as development proceeds. User interface elements are always refined as they are used. Also, changing business needs will change requirements over time.

Figure 2-1: Database and Application Development Lifecycle





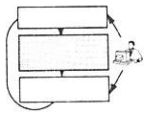
Design Phase

The design phase begins with requirements definition. In consultation with knowledgeable end users, define the functional specifications for the database and applications. Determine which aspects of the functional requirements will be implemented in the database design and which aspects will be implemented in the applications.

Some functions can be performed either by the database or by applications; for example, a complex mathematical transform function could be performed either by a client application program module or by a user-defined function on the server. Try to determine what the most likely deployment configuration will be and allocate the function accordingly. For example, if the client platforms are expected to be low-end desktop PCs, and the server platform is expected to be a high-end workstation, consider performing calculation-intensive functions on the server with UDFs or stored procedures. If the hardware configuration changes, then it is possible to move the function between client and server in a later iteration.

Database design is a complex task that begins with data modeling. Data modeling involves analyzing organizational data and processes and mapping these to database components. For complex data modeling, you may want to use CASE tools to assist in building entity-relationship models. The details of data modeling and database design are based on relational theory and are beyond the scope of this documentation. For a complete discussion of relational theory, refer to any of the authoritative texts on this subject.

Application design is based on the previously defined requirements. From there, it becomes a largely conceptual task of mapping the requirements to specific application development tools. If you are upsizing your applications, then much of the application design may already be done, or may only require minor modification. If you are designing a new application, you should evaluate the variety of application development tools offered by Borland to choose the one or ones that best meet your needs.



Implementation Phase

In the implementation phase you actually build the database and applications conceived in the design phase.

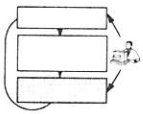
For the database, this phase involves creating the database itself and its constituent metadata: tables, views, columns, triggers, stored procedures, and so on. InterBase provides an interactive SQL data definition tool called **isql**, available as Windows ISQL or command-line **isql**. For an in-depth explanation of data

definition, see the online Help or the *Data Definition Guide*. For a description of using Windows ISQL, see Chapter 3: "Using Windows ISQL."

For the application, this phase involves actually building the application with the chosen tools. Upsizing applications will already be in place, but may require modification or extension. Custom applications may be created with Borland C++, using either the InterBase API, or embedded SQL and dynamic SQL (DSQL). For a full explanation of creating embedded SQL applications, see the online Help or the *Programmer's Guide*. For a complete discussion of creating API applications, see the online Help or the *API Guide*.

Note Local InterBase provided with Borland client tools does not permit native access directly through the InterBase API, but through the Borland Database Engine (BDE). The InterBase API is provided only with the InterBase Server.

After the database has been defined, it must be populated with data. Data can be inserted into InterBase tables directly using **isql**. You can also write a simple application to load data from existing files. Often, the data must be migrated from its present format into InterBase tables.



Deployment Phase

In the deployment phase, the database/application is put to the test. The product created in the implementation phase is handed over to end users. Inevitably, there will be suggestions and requests from users for additional or modified features. As these features are implemented in subsequent iterations, the database/application moves closer to completion. Sometimes it may be advisable to deploy a prototype database/application to help direct subsequent design and development.

During this phase, developers must repair programmatic bugs, respond to user suggestions, and add new features. Since the ultimate judges of an application's efficacy are its users, application developers must be prepared to incorporate changes to applications arising from changing needs in the real world, as well as general enhancements. Often application changes will require changes to the database, and conversely, changes to the database can require application changes.

At some point, a deployed application moves from being a test base to a production database/application, which must then be maintained. The person who maintains a database is known as the *database administrator* (DBA). The DBA controls database security, and performs routine maintenance tasks such as *sweeping* outdated records and transaction recovery. The DBA also performs database backup and restoration as needed. InterBase provides a database administration tool called Server Manager, which runs on Windows 95 and Windows NT. Inter-

Base also provides command-line DBA utilities the run on InterBase Servers and perform the same DBA tasks.

Every InterBase server comes with a SYSDBA account for the use of the DBA. The SYSDBA is the only user who can add new users and change passwords to an InterBase server.

Building Databases

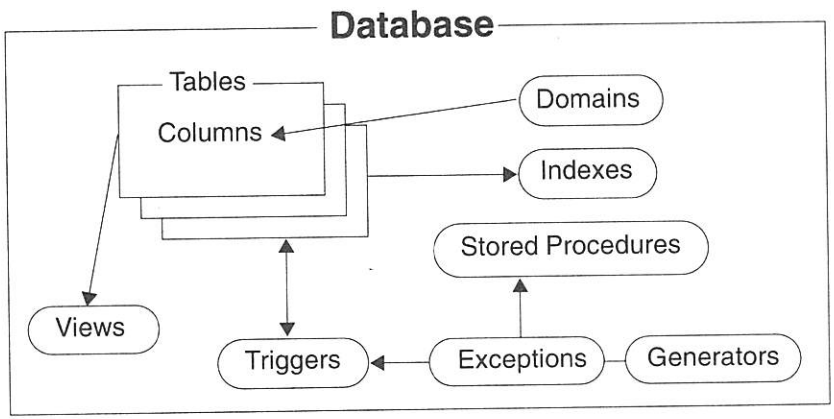
To create a database and its components, InterBase uses an implementation of SQL that conforms to the ANSI SQL-89 entry-level standard and follows SQL-92 and SQL3 beta specifications for advanced features.

Building a database involves *defining the data*. For this purpose, InterBase provides a set of statements called the Data Definition Language (DDL).

A database consists of a variety of database objects, such as tables, views, domains, stored procedures, triggers, and so on. *Database objects* contain all the information about the structure of the database and the data. Because they encapsulate information about the data, database objects are sometimes referred to as *metadata*.

An InterBase database is a single file comprising all the metadata and data in the database. To create a new InterBase database, use Windows ISQL or command-line ISQL, see Chapter 3: "Using Windows ISQL." If you are unfamiliar with creating databases or have never used InterBase, the SQL tutorial provided in online Help form is a good place to start.

Figure 2-1: Database Objects



Tables

Relational databases store all their data in tables. A *table* is a data structure consisting of an unordered set of horizontal *rows*, each containing the same number of vertical *columns*. The intersection of an individual row and column is a *field* that contains a specific piece of information. Much of the power of relational databases comes from defining the relationships among the tables.

InterBase stores information about metadata in special tables, called *system tables*. System tables have predefined columns that store information about the type of metadata in that table. All system tables begin with "RDB$". An example of a system table is RDB$RELATIONS, which stores information about each table in the database.

System tables have the same structure as user-defined tables and are stored in the same database as the user-defined tables. Because the metadata, user-defined tables, and data are all stored in the same database file, each database is a complete unit and can be easily transported between machines.

System tables can be modified like any other database tables. Unless you understand all the interrelationships between the system tables, however, modifying them directly may adversely affect other system tables and disrupt your database. For more information about system tables, see the online Help or the *Language Reference*.

Columns

Creating a table mainly involves defining the columns in the table. The main attributes of a column include:

- The name of the column
- Data type of the column or the domain on which it is based
- Whether or not the column is allowed to be NULL
- Optional referential integrity constraints

Data Types

Data is stored in a predefined format called a data type. Data types can be classified into four categories: numeric, character, date, and BLOB. Numeric data types handle everything from integers to double-precision floating point values. Character data types hold strings of text. Date data types are used for storing

date and time values. InterBase also supports arrays of these standard data types.

While numeric, character, and date are standard data types, the BLOB data type and arrays of standard data types deserve special mention.

Numeric Data Types

Numeric data types are: SMALLINT, INTEGER, FLOAT, DOUBLE PRECISION, NUMERIC, and DECIMAL. Most of these correspond in size and precision to similar data types in C. For example, SMALLINT typically corresponds to a short in C, and DOUBLE PRECISION corresponds to a double. When comparing or assigning values of different numeric types, InterBase handles many conversions automatically. Others can be coerced using the CAST() function.

Character Data Types

Character data types are CHAR and VARCHAR. They allow strings of multiple characters to be stored in a column. CHAR and VARCHAR differ in the way extra characters are treated. The CHAR data type uses all characters up to the end of the array, but the VARCHAR data type is significant only to the first NULL character.

Date Data Types

The DATE data type is used to store date and time values. InterBase handles assignment and comparison between strings and dates. String values representing dates can be in a variety of formats, such as "12-1-94" and "December 1, 1994". Certain date constants are also supported, such as "TODAY" and "TOMORROW".

BLOB Data Types

InterBase supports a *binary large object* (BLOB) data type, which can hold data of unlimited size. The BLOB is an extension of the standard relational model, which ordinarily provides only for data types of fixed width.

The BLOB data type is analogous to a flat file because BLOB data can be stored in any format (for example, binary or ASCII). A BLOB, however, is not a separate file. BLOB data is stored in the database with all other data. Because BLOB columns often contain large, variable amounts of data, BLOB columns are stored and retrieved in segments.

Conversion of BLOB data to other data types in InterBase is not directly supported, but on some platforms, BLOB filters can translate BLOB data from one BLOB format to another.

Arrays of Data Types

InterBase supports arrays of all data types except BLOB. An *array* is a collection of values, or elements, each of the same data type. Individual array elements, blocks of contiguous elements, or the entire array can be accessed using standard SQL statements and API calls.

An array in InterBase can be up to 16 dimensions. Because InterBase arrays are multidimensional, you can store arrays as a whole in a single field, making accessing and retrieval fast and simple. An element of array data is referenced through the use of coordinates, or offsets, into the array.

Domains

In addition to explicitly stating the data type of columns, InterBase allows global column definitions, or *domains*, upon which column definitions can be based. A domain specifies a data type, and a set of column attributes and constraints. Subsequent table definitions can use the domain to define columns.

Referential Integrity Constraints

InterBase allows you to define referential integrity rules for a column, called *referential integrity constraints*. Integrity constraints govern column-to-table and table-to-table relationships and validate data entries. They are implemented through primary keys, foreign keys, and check constraints. Basically, a *primary key* is a column (or group of columns) that uniquely identifies a row in a table. A *foreign key* is a column whose value must match a value of a column in another table. A *check constraint* limits data entry to a specific range or set of values.

For example, an EMPLOYEE table could be defined to have a foreign key column named DEPT_NO that is defined to match the department number column in a DEPARTMENT table. This would ensure that each employee in the EMPLOYEE table is assigned to an existing department in the DEPARTMENT table.

For more information about referential integrity, see the online Help or the *Data Definition Guide*.

Indexes

Indexes are mechanisms for improving the speed of data retrieval. An index identifies columns that can be used to retrieve and sort rows efficiently in the table. It provides a means to scan only a specific subset of the rows in a table, improving the speed of data access.

InterBase automatically defines unique indexes for a table's PRIMARY KEY and FOREIGN KEY constraints. For more information about indexes, see the online Help or the *Data Definition Guide*.

Views

A *view* is a virtual table that is not physically stored in the database, but appears exactly like a "real" table. A view can contain data from one or more tables or other views and is used to store often-used queries or query sets in a database.

Views can also provide a limited means of security, because they can provide users access to a subset of available data while hiding other related and sensitive data. For more information about views, see the online Help or the *Data Definition Guide*.

Stored Procedures

A *stored procedure* is a self-contained program written in InterBase procedure and trigger language, an extension of SQL. Stored procedures are part of a database's metadata. Stored procedures can receive input parameters from and return values to applications, and can be executed explicitly from applications or substituted for a table name in a SELECT statement.

Stored procedures provide:

- Modular design: stored procedures can be shared by applications that access the same database, eliminating duplicate code and reducing the size of applications.
- Streamlined maintenance: when a procedure is updated, the changes are automatically reflected in all applications that use it without the need to recompile and relink them. They are compiled and optimized only once for each client.
- Improved performance: especially for remote client access. Stored procedures are executed by the server, not the client, which reduces network traffic.

For information about stored procedures, see the online Help or the *Data Definition Guide*.

Triggers

A *trigger* is a self-contained routine associated with a table or view that automatically performs an action when a row in the table or view is inserted, updated, or deleted.

Triggers can provide:

- *Automatic enforcement of data restrictions* to ensure that users enter only valid values into columns.
- Reduced application maintenance, because changes to a trigger are automatically reflected in all applications that use the associated table without the need to recompile and relink them.
- Automatic logging of changes to tables. An application can keep a running log of changes with a trigger that fires whenever a table is modified.
- Event alerters in triggers can automatically notify applications of changes to the database.

When a trigger is invoked, it has immediate access to data being stored, modified, or erased. The trigger may also access data in other tables. Using the available data, you can design the trigger to:

- Abort an operation, possibly with an error message.
- Set values in the accessed record.
- Insert, update, or delete rows in other tables.
- Signal that an event has occurred using an event alerter.

Event Alerters

An *event alerter* is a message passed from the database to an application, signalling the occurrence of a database event. An *event* is a particular modification or set of modifications to the database that is important to the application.

An event alerter may be sent from an application directly or from a trigger. If a trigger is used, you can define the event as notification of an insert, update, or delete within a particular table. Use trigger language to define the specific conditions for the event. Thus, an event could be as general as "a row has been stored into table X" or more specific, such as "the price of stock Y has exceeded its assessed value."

The application specifies which event alerters it wants to receive by registering interest in the event to the database. The database will send an event alerter to the application when the specified event occurs.

An application may cease processing and wait for any number of events to occur. Alternatively, the application may receive notification of events asynchronously, allowing other work to be performed until interrupted by the event alerter.

Events:

- Reduce the overhead and network traffic that polling can cause.
- Provide more real-time information than periodic polling.
- Provide timely notification of significant changes to interested users in a heterogenous environment.
- Reduce application code maintenance, because the same events are usable by multiple applications.

User-defined Functions

A *user-defined function* (UDF) is a host-language program that runs on the server and is called by database applications. Once the UDF is declared in a database, any application that uses that database may use the function when accessing data. UDFs allow you to modularize an application by separating it into more reusable and manageable units.

UDFs are used for performing customized, often-used tasks in applications, for statistical computations, or for data conversions that are not directly supported by InterBase.

BLOB Filters

A *BLOB filter* is a specialized function that is used to convert BLOB data from one format to another for display or storage purposes. Since the format of the data stored in a BLOB is user-defined, InterBase does not know how to process a BLOB. You define a BLOB subtype for each BLOB format to be used in the application, such as text or graphics. A BLOB filter can then be defined for conversion from one subtype to another.

A BLOB filter is similar to a user-defined function, except that it has predetermined parameters that pass information about the BLOB being converted. The executable BLOB filter routine is made available to InterBase in the same way as a user-defined function.

If BLOB data is often converted from one format to another, a BLOB filter can save:

- Storage space. When storing a BLOB in the database, a BLOB filter could be used to compress the BLOB before it is written to disk. A corresponding filter could then be used to decompress the data when retrieving the BLOB.
- Processing time. When retrieving a BLOB, a filter could be used to retrieve only the desired segments. This reduces the number of calls to the database.
- Network traffic. A BLOB filter could be used to copy a BLOB directly from one format to another. Otherwise, the application would have to retrieve the entire BLOB directly from a remote location, convert it, and pass it back.

InterBase supplies two BLOB filters: a text filter, for displaying straight text files, and a binary filter, which passes BLOB data unchanged. For more information about BLOB filters, see the online Help or the *Programmer's Guide*.

Application Development

Once you create and populate a database, you can access the information through an application. If you use one of Borland's client tools, you can access information through your existing application. You can also design and implement a new application by embedding SQL statements or API calls in an application written in a programming language such as C or C++.

Borland Client Tools Applications

Client/server versions of Borland client tools such as Delphi, Borland C++, Paradox, and Visual dBASE can access InterBase databases using Borland SQL Links. Server query reporting is built into the client tool providing Windows application support. This enables you to build sophisticated, user-friendly database applications with minimal programming effort.

Embedded Applications

You can write your own application using C or C++, or another programming language, and embed SQL statements in the code. You then preprocess the application using **gpre**, the InterBase application development preprocessor.

gpre takes SQL embedded in a host language such as C or C++, and generates a file that can be compiled by the host-language compiler.

Applications can be created with either predefined or dynamic queries and require the development version of the InterBase Server. For more information about compiling, see the online Help or the *Programmer's Guide*.

Predefined Database Queries

Some applications are designed with a specific set of requests or tasks in mind. These applications can specify exact SQL statements in the code for preprocessing and compiling, since they do not change at run time. For more information about preprocessing and compiling, see the online Help or the *Programmer's Guide*.

Dynamic Applications

Some applications may need to be able to handle *ad hoc* SQL statements entered by users at run time; for example, allowing a user to select a record by specifying criteria to a query. This requires that the program construct the query based on user input.

InterBase uses Dynamic SQL (DSQL) for generating dynamic queries. Using DSQL, you can build templates in advance for as much or as little of the query as appropriate. At run time, supply the missing syntax and a buffer for passing data back and forth. For more information about DSQL, see the online Help or the *Programmer's Guide*.

API Applications

The InterBase API is a set of low-level functions that enables applications to construct and send SQL statements to the InterBase engine and receive results back. All database work can be performed through calls to the API.

While programming with the API requires an application developer to build and populate underlying structures commonly hidden at the SQL level, the API is ultimately more powerful and flexible. Applications built using API calls offer the following advantages:

- Control over memory allocation
- No need for precompiling
- Control over error messages

- Direct control over transactions

API functions can be divided into seven categories, according to the object they operate on:

- Database attach and detach
- Transaction start, prepare, commit, and rollback
- BLOB calls
- Array calls
- Event handling
- Informational calls
- Date and integer conversions

For more information about the InterBase API, see the online Help or the *API Guide*.

Multi-database Applications

Unlike many relational databases, InterBase applications can use multiple databases at the same time. Most applications use only one database, but others may need to use several databases which have the same or a different structure.

For example, each project in a department may have a database to keep track of its progress, and the department may need to produce a report of all the active projects. Another example where more than one database would be used is where sensitive data is combined with generally available data. One database could be created for the sensitive data with access to it limited to a few users, while the other database could be open to a larger group of users.

With InterBase you can open and access any number of databases at the same time. You cannot join tables from separate databases, but you can use cursors to combine information. For information about multi-database applications programming, see the online Help or the *Programmer's Guide*.

Communications Protocols

InterBase supports several different communications protocols, allowing an application to connect to a database through:

- TCP/IP
- Named Pipes/NetBEUI

If a connection can be made from one node to another via these protocols, then an application can connect to a database on the remote node by communication with a InterBase server on that node. All communication is handled transparently by InterBase, so that the connection appears as if were local.

PART 2

Data Definition Tools

Part 2 explains Windows ISQL, InterBase's interactive SQL tool.

Chapter 3: "Using Windows ISQL" describes how to use Windows ISQL. Windows ISQL can be used to define, query, and manipulate data on InterBase servers.

Chapter 4: "Using ISQL Script Files" describes how to run an ISQL script file from Windows ISQL, and provides details on ISQL commands that can be used in scripts.

CHAPTER 3

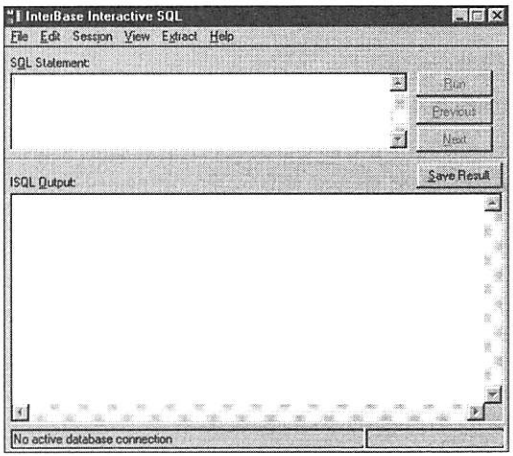
Using Windows ISQL

This chapter describes how to use Windows ISQL, InterBase's interactive SQL tool. Windows ISQL is used to define, query, and manipulate data on InterBase servers.

Starting and Exiting Windows ISQL

Wisql32.exe

To start Windows ISQL, double-click on the Windows ISQL icon. The ISQL window will open:



The ISQL window can also be opened from the Server Manager by choosing Tasks | Interactive SQL or clicking on the corresponding ToolBar button. Windows ISQL will then be connected to Server Manager's current database (if any).

The ISQL Window

The Interactive SQL window consists of a menu bar with pull-down menus, the SQL Statement area, the ISQL Output area, control buttons, and a status bar at the bottom of the window.

The ISQL menus are:

- File menu—contains commands to connect to, create, drop, and disconnect from a database, execute an SQL script file, save results and the session to a file, commit and roll back work, and exit ISQL.
- Session menu—contains statements to set basic and advanced ISQL settings and display ISQL settings and version.
- View menu—contains a command to view metadata.
- Extract menu—contains commands to extract metadata for databases, tables, and views.
- Help menu—provides online help.

The SQL Statement area is where you type an SQL statement to be executed. It scrolls vertically.

The ISQL Output area is where the results of the SQL statements are displayed. It scrolls both vertically and horizontally.

The three buttons to the right of the SQL Statement area, Run, Previous, and Next, are used to execute SQL statements interactively and select statements in the SQL command history. For more information about using these buttons, see "Executing SQL Interactively," in this chapter. The button above the ISQL Output area labeled Save Result opens a dialog box in which you can enter a file name to which to save the results of the last SQL statement executed.

The status bar at the bottom of the ISQL window shows the name of the database to which Windows ISQL is connected or "No active database connection" if it is not connected to a database.

To use Windows ISQL, you must either create a new database or connect to an existing database.

Getting Help

Windows ISQL provides a full online help system. Choose one of the items on the Help menu or click on a Help button in a dialog box to get help.

Exiting Windows ISQL

To exit Windows ISQL, choose File | Exit. This will close the connection to the current database (if any) and exit Windows ISQL. Any uncommitted changes to the database will be rolled back.

Temporary Files

Windows ISQL creates temporary files used during a session to store information such as the command history, output file names, and so on. These files are named ISQL_AA.*xx*, where *xx* is a pair of sequential generated letters. The files are stored in the directory specified by the TMP environment variable, or if that is not defined, the working directory, or if that is not defined, then the WINDOWS directory.

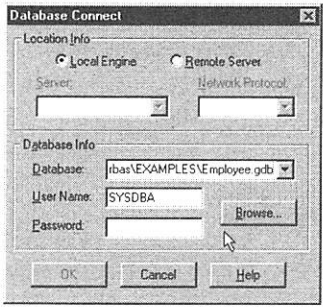
To avoid cluttering the WINDOWS directory with temporary files, specify a directory in which to store them by defining TMP or by defining a working directory for Windows ISQL (by choosing File | Properties in Program Manager).

When you exit, Windows ISQL will delete these temporary files. If Windows ISQL abnormally terminates (for example, due to a power failure), then these files will remain and may be freely deleted without any adverse effects. You should not delete any of these temporary files while Windows ISQL is running, because they may be used in the current session.

Connecting to an Existing Database

Choose File | Connect to Database... to connect to an existing database. If Windows ISQL is currently connected to a database, the connection will be closed; a dialog box will prompt you to commit changes to it (if there are any). If you choose No, then all database changes since the last commit will be rolled back and the connection will be closed. If you choose Yes, then database changes will be committed.

The Database Connect dialog box will open:



The Database Connection dialog box enables you to connect to databases on a local or remote server. If an InterBase local database server is running on your PC, the Local Engine button will be enabled.

To connect to a local database server, choose Local Server if not enabled and provide the name of the database (including full volume and directory path), a valid username, and password. You can select a database from the drop-down list of previously used databases or use the Browse button to select other databases.

To connect to a remote database server, choose Remote Server and select a network protocol from the drop-down list. Information about network protocols is provided in Help.You must provide the name of the database (including full volume and directory path) and a valid username and password. You can select a database from the drop-down list of previously used databases.

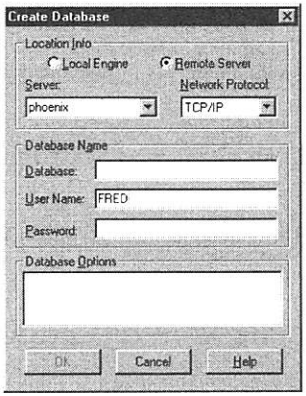
Important For access to InterBase databases on a server, a valid user name and password is required. This is validated against the security database, ISC4.GDB. Every InterBase Server comes with SYSDBA user and password "masterkey". SYSDBA must authorize all other users on a server. For servers that require secure databases, it is strongly recommended that the database administrator change the "masterkey" password for SQSDBA as soon as possible and assign authorized users to the database. For information about database access, see Chapter 7: "Server Logon and Database Access."

Creating a New Database

To create a new database and connect to it, choose File | Create Database.... If currently connected to a database, a dialog box will prompt you to commit changes to it (if any). If you choose No, then all database changes since the last

commit will be rolled back. If you choose Yes, then database changes will be committed.

The Create Database dialog box will open:



The Create Database dialog box enables you to create and connect to databases on a local or remote server.

To create a local database, choose Local Server if not enabled and provide the name of the database (including full volume and directory path), a valid user-name, and password.

To create a remote database, choose Remote Server and select a network protocol from the drop-down list. Information about network protocols is provided in the online Help. You must provide the name of the database (including full volume and directory path) and a valid username and password.

In the Database Options area, enter any additional options of the CREATE DATABASE statement, such as PAGE_SIZE, DEFAULT CHARACTER SET, or secondary files. For a complete list of CREATE DATABASE options, see the online Help or the *Language Reference*. To create a basic database without any options, leave the Database Options area blank.

Note Primary database files must reside on a local drive. Secondary files can reside on either local or on remote drives.

Choose OK to create the database. ISQL will then create the database on the specified server and connect to the database.

For more information about creating databases, see the online Help or the *Data Definition Guide*.

Dropping a Database

Dropping a database deletes the database to which ISQL is currently connected, removing both data and metadata. To drop the current database, choose File | Drop Database.... A dialog box will ask you to confirm that you want to delete the database. A database can be dropped only by its creator or the SYSDBA user.

A dropped database is removed from the list of databases maintained in INTERBAS.INI.

Caution Dropping a database deletes all data and metadata in the database.

Disconnecting from a Database

To disconnect from the database to which Windows ISQL is connected, choose File | Disconnect from Database.... A dialog box will open to confirm that you want to disconnect. If there are any uncommitted database changes, you will be prompted to commit them before disconnecting.

Executing SQL Statements

In Windows ISQL, you can execute SQL statements:

- Interactively, one statement at a time.
- From a file containing an SQL script.

Executing SQL Interactively

To execute an SQL statement interactively, type it in the SQL Statement area and choose Run or press **Alt+R**. The statement will be echoed, and up to 32K of the results displayed in the ISQL Output area. Any output beyond 32K will be scrolled out of the ISQL Output Area.

Tip You can copy text from other Windows applications (such as the Notepad text editor) and paste it into the SQL Statement area with **Ctrl+V**. You can also copy statements from the ISQL Output area by highlighting them and pressing **Ctrl+C**. You can then paste them into the SQL Statement area with **Ctrl+V**. The cut-and-paste method is also a convenient way to use the online SQL tutorial provided in the online Help.

When an SQL statement is executed (whether successfully or not), it becomes part of the ISQL *command history*, a sequential list of SQL statements entered in the current session. The *current statement* is the statement displayed in the SQL Statement area.

The three buttons to the right of the SQL Statement area are:

- Run: executes the current statement. The resultant output is displayed in the ISQL Output area. This button is dimmed if there is no active database connection.
- Previous: recalls the previous SQL statement in the command history, making it the current statement. When the current statement is the first statement in the command history, this button is dimmed and you may not choose it.
- Next: recalls the next SQL statement in the command history, making it the current statement. When the current statement is the last statement in the command history, this button is dimmed and you may not choose it.

As an alternative to these buttons, use the hot keys **Alt+R**, **Alt+P**, and **Alt+N**, respectively. The hot key for each button is underlined in its label.

Legal Statements

You can execute interactively any SQL statement identified as "available in ISQL" in the online Help or *Language Reference*.

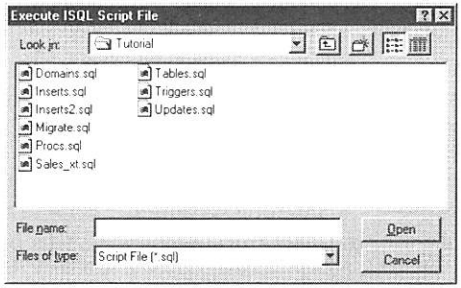
Note The SET NAMES statement cannot be entered in the SQL Statement area. To change the active character set, choose Session | Advanced Settings... and select the desired character set in the Advanced Set Options dialog box.

Transaction names may not be used with SET TRANSACTION statement. Each SQL statement optionally may be terminated by a semicolon (;).

SQL script files can include statements that are not legal to enter interactively. For example, most of the SET statements such as SET LIST or SET TERM can be used in scripts, but cannot be entered interactively. Use the Session menu items to perform the corresponding functions for an interactive session.

Executing an ISQL Script File

To execute a file containing SQL statements, choose File | Run ISQL Script.... The following dialog box will appear in Windows 95 (Windows NT uses a different file dialog):



Choose the file containing the script to run. If you have made uncommitted changes to the database, you will be prompted to commit or roll back the work. Then, a dialog box will appear asking "Save Output to a File?" If you choose Yes, then another dialog box will appear enabling you to specify an output file. If you choose No, then the results will be displayed in the ISQL Output area. If you choose Cancel, then the operation is canceled.

After Windows ISQL finishes executing a script file, a summary dialog will appear indicating if there were any errors. If there were errors, then an error message will appear in the ISQL Output Area (or output file) after each statement that caused the error.

Every ISQL script file must begin with either a CREATE DATABASE statement or a CONNECT statement (including username and password) to specify the database on which the script file operates. For more information about these SQL statements, see the online Help or the *Language Reference*.

Statements executed in a script file do not become part of the command history.

Committing and Rolling Back Work

Changes to the database from data definition (DDL) statements—for example, CREATE and ALTER statements—are automatically committed by default. To turn off automatic commit of DDL, choose Session | Basic ISQL Settings... and click off the Auto Commit DDL check box.

Changes made to the database by data manipulation (DML) statements—for example INSERT and UPDATE—are not permanent until they are committed. Commit changes by choosing File | Commit Work.

To undo all database changes from DML statements since the last commit, choose File | Rollback Work.

Saving Results to a File

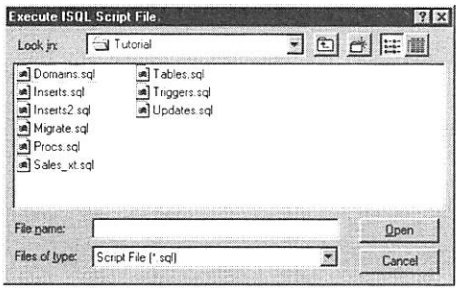
Windows ISQL enables you to save to a file:

- The output of the last SQL statement executed.
- SQL statements entered in the current session.

Saving ISQL Output

To save to a file the results of the last SQL statement executed, choose File | Save Result to File... or click on the Save Result button in the ISQL window. You can also use the hot key **Alt+S**.

The following dialog box will appear in Windows 95 (Windows NT uses a different dialog):

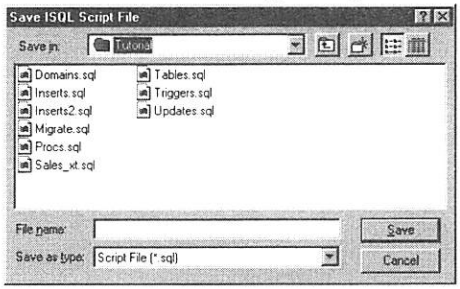


Choose the desired directory and file name or type the file name in the text field, and choose OK. The output from the last successful statement and the statement itself will be saved to the named text file.

If you run an SQL script, and then choose File | Save Result to File..., then all the commands in the script file and their results will be saved to the output file. If command display has been turned off in a script with SET ECHO OFF, then SQL statements in the script will not be saved to the file.

Saving the Session

To save the SQL statements entered in the current session to a text file, choose File | Save Session to a File.... The following dialog box will appear:



Choose the desired directory and file name or type the file name, and click OK to save the SQL statements to the file.

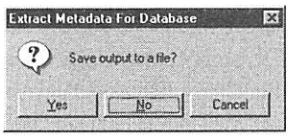
Only the SQL statements entered in the current session, not the output, will be saved to the specified file.

Extracting Metadata

Windows ISQL enables you to extract metadata for the entire database and for a specific table or view.

Extracting Database Metadata

To extract data definition statements (metadata) from a database to a file, choose Extract | SQL Metadata for Database.... The following dialog box will open:



If you choose Yes, then another dialog box will open, enabling you to enter the name of the file to which to extract the metadata. If you choose No, then the metadata will be displayed to the ISQL Output area only. If you choose Cancel, then the operation will be canceled.

This command does not extract:

- Generators.
- Code of external functions or filters, because that code is not part of the database. The declarations to the database (with DECLARE EXTERNAL FUNCTION and DECLARE FILTER) are extracted.
- System tables, system views, and system triggers.

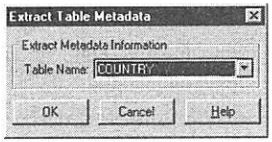
This command extracts metadata in the following order:

Table 3-1: Order of Metadata Extraction

Metadata	Comments
Database	Extracts database with default character set and PAGE_SIZE.
Domains	
Tables	
BLOB data types and known subtypes	
NULL and default values	
PRIMARY KEY constraints	
CHECK constraints	
FOREIGN KEY constraints	Must be added after tables by ALTER TABLE to avoid tables referenced before being created.
Indexes	Only for tables extracted, except triggers from referential or unique constraints.
Views WITH CHECK OPTION	
Stored procedures	In the extracted DDL, stored procedures are shown with no body in CREATE PROCEDURE and then ALTER PROCEDURE to add the text of the procedure body.
Triggers	Does not extract triggers from CHECK constraints.
GRANTs	From RDB$USER_PRIVILEGES table.

Extracting Table Metadata

To extract metadata for a single table, choose Extract | SQL Metadata for Table.... The following dialog box will open:



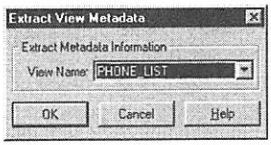
Click on the arrow to the right of the Table Name field to see a drop-down list of tables in the database. Select a table, then choose OK to extract metadata from that table.

Another dialog box will open, asking whether to save output to a file. Choose Yes to save the metadata to a text file, No to display the metadata to the Output area only, or Cancel to cancel the operation.

If there are no tables in the database, then the menu item will be dimmed, and you cannot select it.

Extracting View Metadata

To extract metadata for a single view, choose Extract | SQL Metadata for View.... The following dialog box will open:



Click on the arrow to the right of the View Name field to see a drop-down list of views in the database. Choose a view, then choose OK to extract metadata from that view.

Another dialog box will open, asking whether to save output to a file. Choose Yes to save the metadata to a text file, No to display the metadata to the Output area only, or Cancel to cancel the operation.

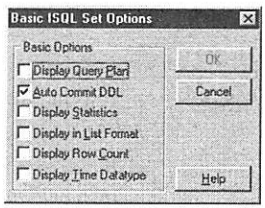
If there are no views defined for the database, then the menu item will be dimmed, and you cannot select it.

Changing Windows ISQL Settings

The Session menu enables you to change ISQL settings for the current session and display information about the database and its metadata.

Basic ISQL Settings

Choose Session | Basic ISQL Settings... to open a dialog box displaying all the basic settings that can be toggled on or off:



Each setting has a corresponding check box. If there is a ✔ in the box, then the setting is on. Otherwise, it is off. Click on the check box or the setting name to toggle the setting.

The following table summarizes basic ISQL settings:

Table 3-2: Basic ISQL Settings

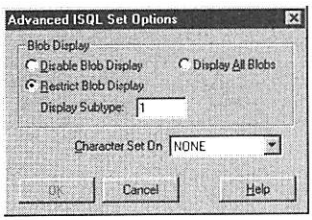
Setting	Behavior when setting is ON
Display Query Plan	Display the query plan chosen by the optimizer when a SELECT is entered. To modify the optimizer plan, use the PLAN option of the SQL SELECT statement.
Auto Commit DDL	Automatically commits DDL (data definition) statements as each statement is entered. This setting is ON by default. If this setting is off, you must explicitly commit DDL statements (with File | Commit Work) to make them permanent.

Table 3-2: Basic ISQL Settings (Continued)

Setting	Behavior when setting is ON
Display Statistics	Displays performance statistics for each statement entered. The following performance statistics appear after the result of each statement: • Number of read or write requests • Number of requests for data or updates that can be serviced in cache • Elapsed time • CPU time • Memory usage • Database page size • Database buffers used
Display in List Format	Displays data in list format, with headings on the left and column values on the right, one row at a time. If this setting is off (default), data is displayed in tabular format with data in rows and columns, which may wrap longer rows.
Display Row Count	Displays the number of rows returned by each SELECT query entered.
Display Time Data Type	Displays the time portion of DATE values. If this setting is OFF, then only the date portion of DATE values is displayed.

Advanced ISQL Settings

There are two advanced ISQL settings: BLOB display and character set choice. Choose Session | Advanced ISQL Settings... to open the following dialog box:



Choose OK to accept all the setting changes, or choose Cancel to cancel setting changes.

BLOB Display

The upper area in the dialog box enables you to determine the display of BLOB data types.

This setting determines the display of BLOB data. SELECT always displays the BLOB ID for columns of BLOB data type. By default, a SELECT will also display actual BLOB data of text subtypes beneath the associated row.

The choices are:

- Disable BLOB Display: Do not display contents of BLOB columns.
- Display ALL BLOBs: Display BLOB data of all subtypes.
- Restrict BLOB Display: Display contents of BLOB columns only for the specified subtype. Use 0 for an unknown subtype; 1 for a text subtype (the default), and other integers for other BLOB subtypes.

Character Set

This setting determines the active character set for strings for subsequent connections to the database. It enables you to override the default character set for a database.

Specify the character set before connecting to the database whose character set you want to specify. For a complete list of character sets recognized by InterBase, see the online Help or the *Language Reference*.

Choice of character set limits possible collation orders to a subset of all available collation orders. Given a character set, a collation order can be specified when data is selected, inserted, or updated in a column.

You can perform the same function in an SQL script with the SET NAMES command. Use SET NAMES before connecting to the database whose character set you want to specify.

Displaying Settings

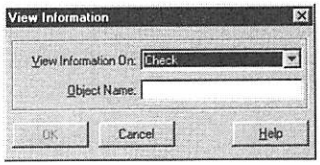
To display all the current ISQL settings, choose Session | Display Settings.... The basic settings and selected advanced settings will be displayed in the ISQL Output area.

Displaying Version Information

To display version information, choose Session | Display Connect Version in the ISQL Output Area. This will display the version of ISQL being used. If connected to a database, this command will also display the versions of the InterBase access method, server, and remote interface.

Displaying Database Information and Metadata

Choose View | Metadata Information... to display database information and metadata. The following dialog box will open:



Select the object type for which to display information, supply any required information in the Object Name text field, and choose OK. Generally, if you do not supply an Object Name, then ISQL will display the names of all objects of the selected type in the database. If you do supply an Object Name, then ISQL will display information about that object.

The following table summarizes the items that can be displayed.

Table 3-3: Metadata Information Items

Item	DIsplays
Check...	Check constraints for the specified table. Specify table name in the Object Name field.
Database	Current database's file name, page size and allocation, and sweep interval. Do not specify an Object Name.
Domain	Names of all domains in the database (with no Object Name). Name and data type of the domain given as Object Name.
Exception	Names of all exceptions in the database, their associated messages, and the names of triggers and stored procedures that use them (with no Object Name). Name and message of exception given as Object Name, and names of triggers and stored procedures that use it.
Generator	Names and current values of all generators in the database (with no Object Name). Name and current value of the generator given as Object Name.

Table 3-3: Metadata Information Items (Continued)

Item	Displays
Grant	Displays permissions for the table or view given as Object Name.
Index	Names of all indexes in the database, their constituent columns, and uniqueness (with no Object Name). Names of all indexes for the table given as Object Name, their constituent columns, and uniqueness. Constituent columns for the index given as Object Name, and the index's uniqueness.
Procedure	Names and dependencies of all stored procedures in the database (with no Object Name). Procedure body, for the procedure given as Object Name, its input parameters, and output parameters.
System	Displays the names of system tables and system views for the current database. Do not specify an Object Name.
Table	Names of all tables in the database (with no Object Name). Columns, data types, PRIMARY KEY, FOREIGN KEY, and CHECK constraints for the table given as Object Name.
Trigger	Names of all triggers in the database and the tables for which they are defined (with no Object Name). Trigger bodies when a table is given as Object Name. Body of the trigger given as Object Name.
View	Names of all views in the database (with no Object Name). Columns, data types, and view source for the view given as Object Name.

CHAPTER 4

Using ISQL Script Files

This chapter describes how to run an ISQL script file from Windows ISQL, and provides details on ISQL commands that can be used in scripts, but not interactively.

Creating and Executing SQL Files

The basic steps for using script files with Windows ISQL are:

- Create the file using a text editor.
- Execute the file with Windows ISQL.
- View output and confirm database changes with Windows ISQL.

Creating an ISQL Script File

You can use any text editor to create an ISQL script file, as long as the final file format is "plain text" (ASCII).

Every ISQL script file must begin with either a CREATE DATABASE statement or a CONNECT statement (including username and password) to specify the database on which the script file operates.

The CONNECT or CREATE statement must contain a complete database file name and directory path.

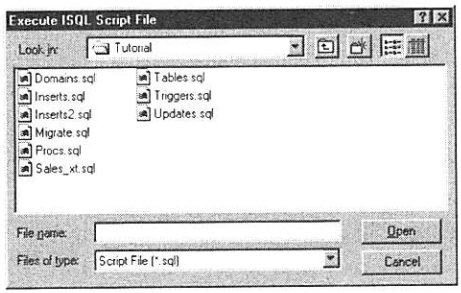
An ISQL script may contain any SQL statements, as described in the *Language Reference*, ISQL SET commands as described in this chapter, and comments. Each SQL statement in a script *must* be terminated by a semicolon (;) or the current terminator if it has been changed with SET TERM.

Each script file should end with either EXIT to commit database changes or QUIT to roll back changes made by the script. If neither is specified, then database changes are committed by default.

For the full syntax of CONNECT and CREATE DATABASE, see the online Help or the *Language Reference.*

Executing an ISQL Script File

To execute a file containing SQL statements, choose File | Execute ISQL Script.... The following dialog box will appear in Windows 95 (Windows NT uses d different dialog):



Enter the path and name of the file and click on OK. A dialog box will appear asking "Save output to a file?" If you choose Yes, then another dialog box will open, enabling you to enter a file name to which to save output. If you choose No, then output and any error messages will be displayed in the SQL Output area.

After Windows ISQL finishes executing a script file, a summary dialog will appear indicating if there were any errors. If there were errors, then an error message will appear in the ISQL Output Area (or output file) after each statement that caused the error.

After a script is executed, all ISQL settings prior to executing it will be restored as well as the previous database connection, if any. Any ISQL SET commands in the script affect only the ISQL session while the script is running.

Committing and Rolling Back Work

Changes to the database from data definition (DDL) statements—for example, CREATE and ALTER statements—are automatically committed by default. This means that other users of the database will see changes as soon as each DDL statement is executed. To turn off automatic commit of DDL in a script, use SET AUTODDL OFF.

Note When creating tables and other database objects, it is good practice to put a COMMIT statement in the ISQL script after each CREATE statement (or group of related statements) to make sure the changes are committed

before creating other database objects and that other users of the database will see the objects immediately.

Changes made to the database by data manipulation (DML) statements—for example INSERT and UPDATE—are not permanent until they are committed. Commit changes in a script with COMMIT. To undo all database changes since the last COMMIT, use ROLLBACK. For the full syntax of COMMIT and ROLLBACK, see the online Help or the *Language Reference*.

Adding Comments

ISQL scripts are commented exactly like C programs:

```
/* comment */
```

A comment may occur on the same line as ISQL commands and may be of any length, as long as it is preceded by "/*" and followed by "*/".

ISQL SET Statements

SET Statements are used to configure the ISQL environment from a script file. Changes to the session setting from SET statements in a script affect the session only while the script is running. After a script completes, the session settings prior to running the script will be restored.

You cannot enter ISQL SET statements interactively in the SQL Statement area. When using ISQL interactively, perform these same functions with the Session menu items. SET GENERATOR and SET TRANSACTION (without a transaction name) are SQL statements and so may be entered interactively. The ISQL SET statements are:

Table 4-1: SET Statements

Statement	Description
SET AUTODDL	Toggles the commit feature for DDL statements.
SET BLOBDISPLAY *n*	Turns on the display of BLOB type *n*. The parameter *n* is required to display BLOB types.
SET COUNT	Toggles the count of selected rows on or off.
SET ECHO	Toggles the display of each command on or off.
SET LIST *string*	Displays columns vertically or horizontally.
SET NAMES	Specifies the active character set.
SET PLAN	Specifies whether or not to display the optimizer's query plan.

Table 4-1: SET Statements (Continued)

Statement	Description
SET STATS	Toggles the display of performance statistics on or off.
SET TERM *string*	Allows you to change to an alternate terminator character(s).
SET TIME	Toggles display of time in DATE values.

By default, all settings are initially OFF except AUTODDL and TIME, and the terminator is a semicolon (;). Each time you start an ISQL session or execute an ISQL script file, settings begin with their default values.

After an ISQL script completes, the settings return to their values before the script was run. So you can modify the settings for interactive use, then change them as needed in an ISQL script, and after running the script they automatically return to their previous configuration.

The statements SET DATABASE, SET GENERATOR, and SET TRANSACTION are not exclusively ISQL commands, so they are not documented in this chapter. For more information about these commands, see the online Help or the *Language Reference*.

SET AUTODDL

Specifies whether DDL statements are committed automatically after being executed or committed only after an explicit COMMIT.

Syntax

```
SET AUTODDL [ON | OFF];
```

Argument	Description
ON	Turns on automatic commitment of DDL (default).
OFF	Turns off automatic commitment of DDL.

Description

SET AUTODDL is used to turn on or off the automatic commitment of data definition language (DDL) statements. By default, DDL statements are automatically committed immediately after they are executed, in a separate transaction. This is the recommended behavior.

If the OFF keyword is specified, auto-commit of DDL is then turned off. In OFF mode, DDL statements can only be committed explicitly through a user's transaction. This mode is useful for database prototyping, because uncommitted changes are easily undone by rolling them back.

SET AUTODDL has a shorthand equivalent, SET AUTO.

Tip The ON and OFF keywords are optional. If they are omitted, SET AUTO switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Examples The following example shows part of an ISQL script that turns off AUTO DDL, creates a table named TEMP, then rolls back the work.

```
...
SET AUTO OFF;
CREATE TABLE TEMP (a INT, b INT);
ROLLBACK;
...
```

This script creates TEMP and then rolls back the statement. If you choose View | Metadata... and select "Tables," the TEMP table will not appear, because its creation was rolled back.

The next script uses the default AUTO DDL ON. It creates the table TEMP and then performs a rollback:

```
...
CREATE TABLE TEMP (a INT, b INT);
ROLLBACK;
...
```

Because DDL is automatically committed, the rollback does not affect the creation of TEMP. If you choose View | Metadata... and select "Tables," you will see the TEMP table.

See Also EXIT, QUIT

SET BLOBDISPLAY

Specifies subtype of BLOB data to display.

Syntax

```
SET BLOBDISPLAY [n | ALL | OFF];
```

Argument	Description
n	Integer specifying the BLOB subtype to display. Use 0 for BLOB data of an unknown subtype; use 1 (default) for BLOB data of a text subtype, and other integer values for other subtypes.
ALL	Display BLOB data of all subtypes.
OFF	Turn off display of BLOB data of all subtypes.

Description

SET BLOBDISPLAY has the following uses:

- To display BLOB data of a particular subtype, use SET BLOBDISPLAY *n*. By default, ISQL displays BLOB data of text subtype (*n* = 1).
- To display BLOB data of all subtypes, use SET BLOBDISPLAY ALL.
- To avoid displaying BLOB data, use SET BLOBDISPLAY OFF. Omitting the OFF keyword has the same effect. Turn BLOB display off to make output easier to read.

In any column containing BLOB data, the actual data does not appear in the column. Instead, the column displays a BLOB ID that represents the data. If SET BLOBDISPLAY is on, data associated with a BLOB ID appears under the row containing the BLOB ID. If SET BLOBDISPLAY is off, the BLOB ID still appears even though its associated data does not.

SET BLOBDISPLAY has a shorthand equivalent, SET BLOB.

To determine the subtype of a BLOB column, use SHOW TABLE.

Examples

The following examples show output from the same SELECT statement. Each example uses a different SET BLOB command to affect how output appears. The first example turns off BLOB display.

```
SET BLOB OFF;
SELECT PROJ_NAME, PROJ_DESC FROM PROJECT;
```

With BLOBDISPLAY OFF, the output shows only the BLOB ID:

```
PROJ_NAME                PROJ_DESC
=====================    =================
Video Database           24:6
DigiPizza                24:8
AutoMap                  24:a
MapBrowser port          24:c
Translator upgrade       24:3b
Marketing project 3      24:3d
```

The next example restores the default by setting BLOBDISPLAY to subtype 1 (text).

```
SET BLOB 1;
SELECT PROJ_NAME, PROJ_DESC FROM PROJECT;
```

Now the contents of the BLOB appear below each BLOB ID:

```
PROJ_NAME                PROJ_DESC
=====================    =================
Video Database           24:6
============================================================
PROJ_DESC:
Design a video data base management system for
controlling on-demand video distribution.

PROJ_NAME                PROJ_DESC
=====================    =================
DigiPizza                24:8
============================================================
PROJ_DESC:
Develop second generation digital pizza maker
with flash-bake heating element and
digital ingredient measuring system.
. . . .
```

See Also BLOBDUMP

SET COUNT

Specifies whether to display number of rows retrieved by queries.

Syntax

```
SET COUNT [ON | OFF];
```

Argument	Description
ON	Turns on display of the "rows returned" message.
OFF	Turns off display of the "rows returned" message (default).

Description By default, when a SELECT statement retrieves rows from a query, no message appears to say how many rows were retrieved.

Use SET COUNT ON to change the default behavior and display the message. To restore the default behavior, use SET COUNT OFF.

Tip The ON and OFF keywords are optional. If they are omitted, SET COUNT switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Example The following example sets COUNT ON to display the number of rows returned by all following queries:

```
SET COUNT ON;
SELECT * FROM COUNTRY
   WHERE CURRENCY LIKE "%FRANC%";
```

The output displayed would then be:

```
COUNTRY                CURRENCY
===============        ==========
SWITZERLAND            SFRANC
FRANCE                 FFRANC
BELGIUM                BFRANC

3 rows returned
```

SET ECHO

Specifies whether commands are displayed to the ISQL Output area before being executed.

Syntax

```
SET ECHO [ON | OFF];
```

Argument	Description
ON	Turns on command echoing (default).
OFF	Turns off command echoing.

Description By default, commands in script files are displayed (echoed) in the ISQL Output area, before being executed. Use SET ECHO OFF to change the default behavior and suppress echoing of commands. This may be useful when sending the output of a script to a file, if you want only the results of the script and not the statements themselves in the output file.

Command echoing is useful if you want to see the commands as well as the results in the ISQL Output area.

Tip The ON and OFF keywords are optional. If they are omitted, SET ECHO switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Example Suppose you execute the following script from Windows ISQL:

```
...
SET ECHO OFF;
SELECT * FROM COUNTRY;
SET ECHO ON;
SELECT * FROM COUNTRY;
EXIT;
```

The output (in a file or the ISQL Output area) will look like this:

```
...
SET ECHO OFF;

COUNTRY      CURRENCY
============ ========
USA          Dollar
England      Pound
...
SELECT * FROM COUNTRY;

COUNTRY      CURRENCY
============ ========
USA          Dollar
England      Pound
...
```

The first SELECT statement is not displayed, because ECHO is OFF. Notice also that the SET ECHO ON statement itself is not displayed, because when it is executed, ECHO is still OFF. After it is executed, however, the second SELECT statement is displayed.

See Also INPUT, OUTPUT

SET LIST

Specifies whether output appears in tabular format or in list format.

Syntax

```
SET LIST [ON | OFF];
```

Argument	Description
ON	Turns on list format for display of output.
OFF	Turns off list format for display of output (default).

Description

By default, when a SELECT statement retrieves rows from a query, the output appears in a tabular format, with data organized in rows and columns.

Use SET LIST ON to change the default behavior and display output in a list format. In list format, data appears one value per line, with column headings appearing as labels. List format is useful when columnar output is too wide to fit nicely on the screen.

Tip The ON and OFF keywords are optional. If they are omitted, SET LIST switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Example

Suppose you execute the following statement in a script file:

```
SELECT JOB_CODE, JOB_GRADE, JOB_COUNTRY, JOB_TITLE FROM JOB
   WHERE JOB_COUNTRY = "Italy";
```

The output will be:

```
JOB_CODE   JOB_GRADE   JOB_COUNTRY   JOB_TITLE
========   =========   ===========   ====================
SRep       4           Italy         Sales Representative
```

Now suppose you precede the SELECT with SET LIST ON:

```
SET LIST ON;
SELECT JOB_CODE, JOB_GRADE, JOB_COUNTRY, JOB_TITLE FROM JOB
   WHERE JOB_COUNTRY = "Italy";
```

The output will then be:

```
JOB_CODE                SRep
JOB_GRADE               4
JOB_COUNTRY             Italy
JOB_TITLE               Sales Representative
```

SET NAMES

Specifies the active character set to use in database transactions.

Syntax

```
SET NAMES [charset];
```

Argument	Description
charset	Name of the active character set. Default: NONE.

Description

SET NAMES specifies the character set to use for subsequent database connections in ISQL. It enables you to override the default character set for a database. To return to using the default character set, use SET NAMES with no argument.

Use SET NAMES before connecting to the database whose character set you want to specify. For a complete list of character sets recognized by InterBase, see the online Help or the *Language Reference*.

Choice of character set limits possible collation orders to a subset of all available collation orders. Given a specific character set, a specific collation order can be specified when data is selected, inserted, or updated in a column.

Example

The following statement at the beginning of a script file indicates to set the active character set to ISO8859_1 for the subsequent database connection:

```
SET NAMES ISO8859_1;
CONNECT "EAGLE:\USR\INTERBASE\EXAMPLES\EMPLOYEE.GDB";
...
```

SET PLAN

Specifies whether to display the optimizer's query plan.

Syntax

```
SET PLAN [ON | OFF];
```

Argument	Description
ON	Turns on display of the optimizer's query plan.
OFF	Turns off display of the optimizer's query plan (default).

Description

By default, when a SELECT statement retrieves rows from a query, ISQL does not display the query plan used to retrieve the data.

Use SET PLAN ON to change the default behavior and display the query optimizer plan. To restore the default behavior, use SET PLAN OFF.

To change the query optimizer plan, use the PLAN clause in the SELECT statement.

Tip The ON and OFF keywords are optional. If they are omitted, SET PLAN switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Example The following example shows part of a script which sets PLAN ON:

```
SET PLAN ON;
SELECT JOB_COUNTRY, MIN_SALARY FROM JOB
   WHERE MIN_SALARY > 50000
      AND JOB_COUNTRY = "France";
```

The output then includes the query optimizer plan used to retrieve the data as well as the results of the query:

```
PLAN (JOB INDEX (RDB$FOREIGN3,MINSALX,MAXSALX))
JOB_COUNTRY              MIN_SALARY
===============          ======================
France                   118200.00
```

SET STATS

Specifies whether to display performance statistics after the results of a query.

Syntax

```
SET STATS [ON | OFF];
```

Argument	Description
ON	Turns on display of performance statistics.
OFF	Turns off display of performance statistics (default).

Description By default, when a SELECT statement retrieves rows from a query, ISQL does not display performance statistics after the results. Use SET STATS ON to change the default behavior and display performance statistics. To restore the default behavior, use SET STATS OFF. Performance statistics include:

- Current memory available, in bytes
- Change in available memory, in bytes
- Maximum memory available, in bytes

- Elapsed time for the operation
- CPU time for the operation
- Number of cache buffers used
- Number of reads requested
- Number of writes requested
- Number of fetches made

Performance statistics can help determine if changes are needed in system resources, database resources, or query optimization.

Tip The ON and OFF keywords are optional. If they are omitted, SET STATS switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Do not confuse SET STATS with the SQL statement SET STATISTICS, which recalculates the selectivity of an index.

Example The following part of a script file turns on display of statistics and then performs a query:

```
SET STATS ON;
SELECT JOB_COUNTRY, MIN_SALARY FROM JOB
   WHERE MIN_SALARY > 50000
      AND JOB_COUNTRY = "France";
```

The output displays the results of the SELECT statement and the performance statistics for the operation:

```
   JOB_COUNTRY              MIN_SALARY
   ===============          ======================
   France                   118200.00

Current memory = 407552
Delta memory = 0
Max memory = 412672
Elapsed time= 0.49 sec
Cpu = 0.06 sec
Buffers = 75
Reads = 3
Writes = 2
Fetches = 441
```

See Also SHOW DATABASE

SET TERM

Specifies which character or characters signal the end of a command.

Syntax

```
SET TERM string;
```

Argument	Description
string	Specifies a character or characters to use in terminating a statement. Default: semicolon (;).

Description

By default, when a line ends with a semicolon, ISQL interprets it as the end of a command. Use SET TERM to change the default behavior and define a new termination character.

SET TERM is typically used with CREATE PROCEDURE or CREATE TRIGGER. Procedures and triggers are defined using a special "procedure and trigger" language in which statements end with a semicolon. If ISQL were to interpret semicolons as statement terminators, then procedures and triggers would execute during their creation, rather than when they are called.

A script file containing CREATE PROCEDURE or CREATE TRIGGER definitions should include one SET TERM command before the definitions and a corresponding SET TERM after the definitions. The beginning SET TERM defines a new termination character; the ending SET TERM restores the semicolon (;) as the default.

Example

The following example shows a text file that uses SET TERM in creating a procedure. The first SET TERM defines "##" as the termination characters. The matching SET TERM restores ";" as the termination character.

```
SET TERM ## ;
CREATE PROCEDURE ADD_EMP_PROJ (EMP_NO SMALLINT, PROJ_ID CHAR(5))
AS
BEGIN
   BEGIN
      INSERT INTO EMPLOYEE_PROJECT (EMP_NO, PROJ_ID)
         VALUES (:emp_no, :proj_id);
   WHEN SQLCODE -530 DO
   EXCEPTION UNKNOWN_EMP_ID;
   END
   RETURN;
END ##
SET TERM ; ##
```

SET TIME

Specifies whether to display the time portion of a DATE value.

Syntax

```
SET TIME [ON | OFF];
```

Argument	Description
ON	Turns on display of time in DATE value.
OFF	Turns off display of time in DATE value (default).

Description

The InterBase DATE data type includes a date portion (including day, month, and year) and a time portion (including hours, minutes, and seconds).

By default, ISQL displays only the date portion of DATE values. SET TIME ON turns on the display of time values. SET TIME OFF turns off the display of time values.

Note The ON and OFF keywords are optional. If they are omitted, the command toggles time display from ON to OFF or OFF to ON.

Example

The following example shows the default display of a DATE data type, which is to display day, month, and year:

```
SELECT HIRE_DATE FROM EMPLOYEE WHERE EMP_NO = 145;
HIRE_DATE
==========
2-MAY-1994
```

This example shows the effects of SET TIME ON, which causes the hours, minutes and seconds to be displayed as well:

```
SET TIME ON;
SELECT HIRE_DATE FROM EMPLOYEE WHERE EMP_NO = 145;
HIRE_DATE
==========
2-MAY-1994 12:25:00
```

PART 3

Managing the InterBase Server

Part 3 describes InterBase Server database administration tasks.

Chapter 5: "Starting and Stopping the Server" describes how to start up and shut down the server.

Chapter 6: "Introduction to Server Manager" provides an overview of the Server Manager, describing what it can do, and its primary windows, menus, and dialog boxes.

Chapter 7: "Server Logon and Database Access" describes how to logon to an InterBase server, connect to a database, add new users, and modify user information.

Chapter 8: "Maintaining a Database" describes how to perform database maintenance with the Server Manager, including shutting down and restarting a database, configuring database sweeping, managing transaction recovery, and validating a database.

Chapter 9: "Backing Up and Restoring a Database" describes how to do database backup and restoration with the Server Manager.

CHAPTER 5

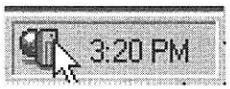
Starting and Stopping the Server

This chapter describes how to start and shut down an InterBase server, run the InterBase Server as an NT service, and view server properties.

Starting InterBase on Windows 95

Local InterBase and InterBase Server must be started prior to enabling database connections.

When you install the Local InterBase server or the InterBase server and reboot your system, the enabled server appears as an icon on the Task Tray, located on the right side of the Task Bar.



To start a server that has been shut down, select the InterBase server from the program group where is has been installed. The enabled server icon appears on the Task Tray.

Selecting Server Startup Options

You can configure how Local InterBase or InterBase server will start by using the Registry Configuration Utility (regcfg.exe). This utility enables you to:

- Start a server each time you start Windows 95. The enabled server appears as an icon on the Task Tray.

- Start a server from the program group where it has been installed by clicking on it. The enabled server then appears as an icon on the Task Tray.

Shutting Down InterBase on Windows 95

To shut down the Local InterBase or InterBase server, right-click on the InterBase server icon located in the Task Tray to display a pop-up menu and select Shutdown. A message appears indicating the number of open database connections. If the message indicates zero (0) active connections, click OK to shut down the server.

If you have open connections, it is recommended that you close them before shutting down the server. You also must close all client applications you are running.

Starting InterBase on Windows NT

The Local InterBase Server and InterBase Server must be started prior to enabling database connections.

When you install the Local InterBase Server and reboot your system, the enabled server appears as an icon on the desktop.

To start a Local InterBase Server that has been shut down, select the Local InterBase server icon from the program group where is has been installed. The enabled server appears as an icon on the desktop.

Selecting Server Startup Options

You can configure InterBase Server services or change its startup options by using the Registry Configuration Utility (regcfg.exe). This utility enables you to:

- Choose how InterBase starts:
  - Start InterBase automatically as a service (recommended). InterBase will start automatically when you start Windows NT.
  - Start InterBase manually as a Service. InterBase must be started using the Services Control Panel or the Registry Configuration Utility (regcfg.exe).

- Start InterBase automatically as an icon. The enabled server appears as an icon on the desktop each time you start Windows NT.
- Start InterBase manually as an icon. The server must be started by clicking on it from the program group where it has been installed. The enabled server appears as an icon on the desktop.

- Choose the process priority. The InterBase server process is a background process that by default runs at normal priority. If there are other processes running at high priority, you may wish to increase the priority of the InterBase process. You can run InterBase in either normal or high-priority.
- Determine the location of the InterBase Server or change the directory for the InterBase license file, message file, and security database.
- Remove InterBase Information from the Registry and Services database.
- Start and Stop NT service. Used to start up and stop the InterBase Server.

If the status of the InterBase Server Service is started, then stop it before changing the status. Once the service has been stopped, you can change how InterBase starts and then restart the NT Service Control.

If you want to change the process priority, you must first stop the InterBase Server Service using the NT Service Control, then choose a different service priority and the InterBase Server Service. Thereafter, the InterBase server processes (database connections) will be started with higher priority.

If InterBase is not getting the CPU share that it needs, and for some reason you do not want to boost the priority of the InterBase process, you can change the default priority of foreground and background processes. Invoke the System application in the Control Panel program group.

In the System dialog box that opens, choose the Tasking... button, then select the option "Foreground and Background Applications Equally Responsive". This will ensure that the InterBase process (that runs in the background) will receive the same priority as foreground processes. Note that all other background processes will have their priority boosted as well.

For more information on foreground and background tasking, see the Microsoft Windows NT documentation.

For more information about InterBase Service configuration on Windows NT, see the Windows NT documentation and the InterBase online Help.

Shutting Down the Server on Windows NT

To shut down the Local InterBase Server or InterBase Server running as an icon, left-click on the server icon to display a pop-up menu and select Shutdown Server from the menu. A message appears indicating the number of open database connections. If the message indicates zero (0) active connections, click OK to shut down the server.

To shut down the InterBase Server running as an NT Service, use the Stop NT Service Control in the Registry Configuration Utility or the NT Service Control.

To use the NT Service Control, highlight the InterBase Server Service and click Stop. If you have the correct user authority, a message box will appear, "Are you sure you want to stop the InterBase Server Service?." Click Yes to stop and No to cancel the operation.

If you have open connections, it is recommended that you close them before shutting down the server. You also must close all client application you are running.

Viewing Server Properties

For information on server properties such as version number and capabilities, left-click the server icon on Windows NT and right-click the server icon on Windows 95 to display a pop-up menu and select Properties from the menu.

CHAPTER 6

Introduction to Server Manager

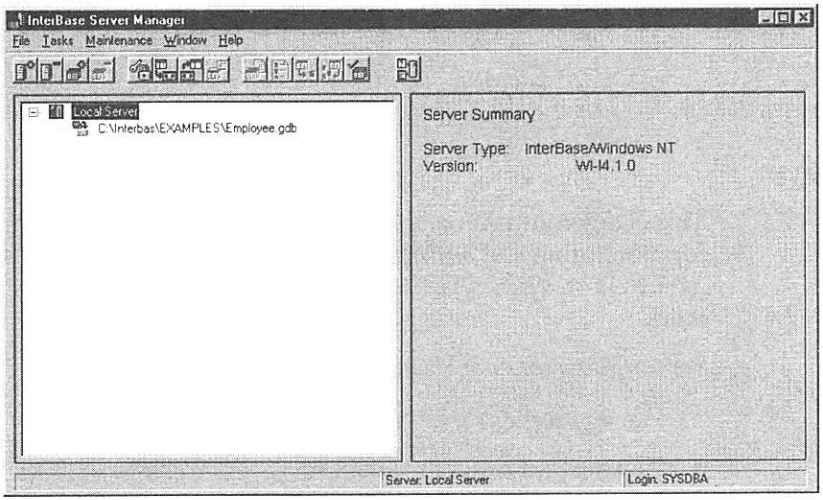
This chapter introduces the InterBase Server Manager, a Windows application for monitoring and administering InterBase databases and servers. Server Manager runs on Windows, but can manage databases on any server on the local network.

Server Manager enables you to:

- Manage server security.
- Back up and restore a database.
- Perform database maintenance, including:
  - Validating the integrity of a database.
  - Sweeping a database.
  - Recovering transactions that are "in limbo."

The Server Manager Window

Start Server Manager by clicking on the Server Manager icon in the InterBase PC Client program group. The Server Manager window will then open:



This window consists of the:

- Menu bar, across the top of the window, containing commands you can choose to perform DBA tasks with Server Manager.
- Toolbar, a row of shortcut buttons for menu commands, just below the menu bar.
- Server/database tree, displayed in the left side of the window below the toolbar, showing the server's name and the databases to which Server Manager is currently connected.
- Summary information area, displayed in the right side of the window below the toolbar. This area displays information about the server or database, depending on which is selected in the server/database tree.
- Status line, which shows the current server and user login and help for menus and the toolbar.

Server Manager Menus

The Server Manager menus are the basic way to perform tasks with Server Manager. There are five pull-down menus:

- File menu: enables you to log in to a server and log out, connect to a database, disconnect from a database, and exit Server Manager.
- Tasks menu: enables you to manage user security, perform backup and restoration, view database and lock manager statistics, and start Windows ISQL.
- Maintenance menu: enables you to view database connections, perform a database sweep, transaction recovery, and database shutdown, restart and validate a database, disable WAL (Write Ahead Log on Netware only), and display database properties.
- Window menu: enables you to close or minimize Server Manager windows and display the toolbar.
- Help menu: provides online Help.

Toolbar

The toolbar is a row of buttons which are shortcuts for menu commands. The toolbar buttons are:

Server login: opens the login dialog box, enabling you to log in to a remote InterBase server. The local server is already connected.

Server logout: enables you to logout from the local server, and disconnect from any databases on that server to which you are currently connected.

Database connect: opens a dialog box that enables you to connect to a database on the current server.

Database disconnect: disconnects Server Manager from the current database.

Configure users: opens the User Configuration dialog box for administering server security.

Database backup: opens the Database Backup dialog box.

Database restore: opens the Database Restore dialog box.

Database statistics: displays relevant information from the database header page.

Database properties: opens the Database Properties dialog box.

Database connections: shows active connections.

Database shutdown: shuts down the database so only the database administrator has access. The database administrator must restart the database before other users have access.

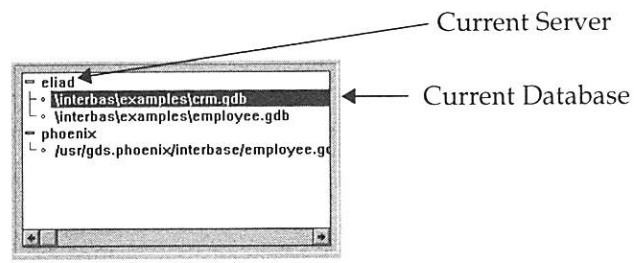
Database restart: restarts a database that has been shut down.

Database validation: opens the Database Validation dialog box.

Start ISQL: opens the Interactive SQL Window and automatically connects to the current database.

Server/Database Tree Pane

When you open the Server Manager window, you must login to a local or remote server and connect to the server's databases to display the server/database tree.



To expand or retract the server/database tree:

Table 6-1: Server/Database Tree Commands

Tasks	Commands
Display a server's databases	• Left- click (plus) + to the left of the server icon • Double-click the server icon • Press the plus (+) key • Press the right arrow key
Retract a server's databases	• Left- click (minus) - to the left of the server icon • Double-click the server icon • Press the minus (-) key • Press the left arrow key

In an expanded tree, click on a database name to highlight it. The highlighted database will be the one upon which Server Manager operates, referred to as the *current database*. When a database is highlighted, the server on which the database resides becomes the *current server*. Any actions of Server Manager then affect that server.

When you right-click the mouse on a server, a pop up menu is displayed listing actions that can be performed on the server:

Table 6-2: Pop-up Menu from Server

Pop-up command	Description
Server Login/Logout	Log in or out of a server
Database Connect	Connect to a database
User Security	Authorize users
Backup	Backup a database
Restore	Restore the database
Lock Manager Statistics	View lock manager statistics report

When you right-click the mouse on a database, a pop up menu is displayed listing actions that can be performed on the database:

Table 6-3: Pop-up Menu from Database

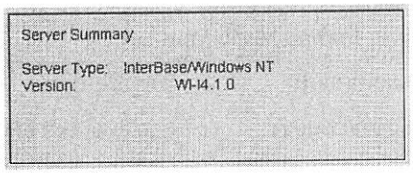
Pop-up command	Description
Database Connect/Disconnect	Connect to or disconnect from a database
Database Statistics	View database header file information

Table 6-3: Pop-up Menu from Database

Pop-up command	Description
Database Connections	View active users on a database
Database Sweep	Perform a database sweep
Transaction Recovery	Recover limbo transactions
Database Shutdown	Shutdown a database
Database Restart	Restart a database
Database Validation	Verify integrity of database structures
Database Properties	View database information, adjust database sweep, enable forced writes.
Database Backup	Back up a database
Interactive SQL	Run Windows ISQL

Summary Information Area

The summary information area in the right side of the Server Manager window displays information about the server or database currently selected in the server/database tree.



Standard Text Display Window

The standard text display window is used to monitor database backup and restoration and to display database and lock manager statistics. Although these operations all display output in a standard text display window, each will have menu commands specific to the particular operation.

The standard text display window contains a menu bar, a toolbar with icons for often-used menu commands, and a scrolling text display area.

CHAPTER 7

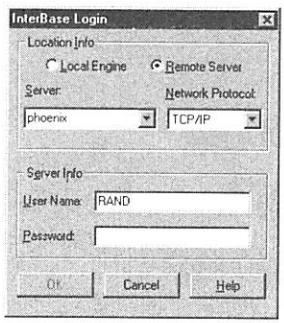
Server Logon and Database Access

Before performing any database administration tasks, you must first login to a server. Once logged in to a server, you may then connect to databases residing on the server. You may switch context from one connected database to another by choosing the desired database in the database/server tree.

Logging in to a Server

Login to a database server by clicking on the Login toolbar button or by choosing File | Server Login.

The following dialog box appears:



To logon to a local or remote server, select Local Engine or Remote Server. Type the name of the server in the Server text field, or click the drop-down arrow for a list of servers to which Server Manager has previously logged in, then choose a server name. Server Manager maintains a list of the last ten servers connected to in the INTERBAS.INI file in the WINDOWS directory.

The InterBase server name is the name of the database server machine. There is not a specific name for the InterBase server process itself. For example, if the server is running on the DEPT1 NT file server, the server name would be DEPT1.

If logging into a remote server, choose a network protocol: either TCP/IP, NetBEUI (used only with Windows NT servers) or SPX/IPX (not supplied with the 32-bit Server Manager).

Enter a username and password in the corresponding text fields. These are the InterBase username and password stored in the security database. For convenience, Server Manager displays the last username and network protocol used; it stores this information in WINDOWS\INTERBAS.INI.

The username is significant to 31 characters and is *not* case-sensitive. Password is significant to eight characters and *is* case-sensitive.

All users must enter their username and password to log in to a server. The password is encrypted for transmission over the network. The username and password are verified against records in the security database. If a matching record is found, the login succeeds.

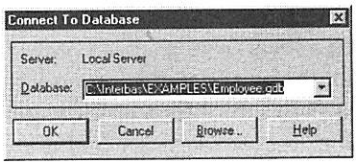
Important Initially, every server has only one authorized user with username SYSDBA. The SYSDBA must log on with password "masterkey" and add all other authorized users. For more information about how to add new users, see "Administering Security," in this chapter.

Click on OK to log in to the server. Once logged in, Server Manager will display the server type and the version of the InterBase server in the summary information area.

Connecting to a Database

Connect to a database by clicking on the Database Connect toolbar button, choosing File | Database Connect, or using the pop-up menu from the server/database tree.

The Connect to Database dialog box will then appear:



The last database connected to on the current server will be shown in the Database text field. Click on the arrow to the right of the text field for a drop-

down list and select a database from the list, or enter a different database name. Choose OK to connect to that database.

If you enter a different name, Server Manager saves the last ten databases to which it connected in the WINDOWS\INTERBAS.INI file. The database file name and path must be appropriate for the type of server it is on.

After connecting to a database, the Server Manager SpeedBar and menus will be active, and any actions you take will apply to the selected database.

Examples

Here are some examples of connecting to databases on various types of servers.

For a Windows NT or Windows 95 server, the database path name must contain the appropriate drive letter designation. For example:

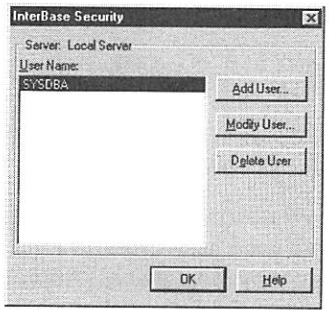
```
D:\USERS\ACCTING\FIN\ACCREC.GDB
```

For a Unix server, you must enter the complete and explicit (not relative) directory path for the database. For example:

```
/usr/accting/fin/accrec.gdb
```

Administering Security

Choose Tasks | User Security or the pop-up menu from the server/database tree to open the InterBase Security dialog box:



The usernames of all authorized users are displayed in the list box.

This dialog box enables you to:

- View the list of authorized users for the server.

- Add a user (if you are SYSDBA).
- Modify users' passwords and optional information (if you are SYSDBA).
- Delete a user (if you are SYSDBA).

The SYSDBA User

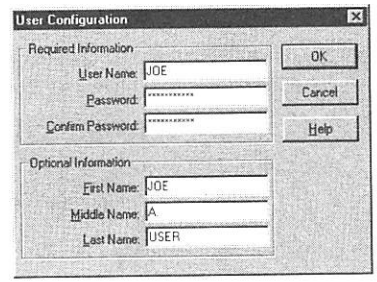
Every InterBase server has a SYSDBA user, with password "masterkey". Initially, SYSDBA is the only authorized user on a server; the SYSDBA must authorize all other users on the server. SYSDBA is a special user that can bypass normal SQL security, and perform special tasks such as database shutdowns.

Only the SYSDBA user can update the security database to add new users, delete users, or modify user configurations. The SYSDBA may authorize a new user by assigning a username and password in the security database.

Important It is strongly recommended to change the password for SYSDBA as soon as possible. Otherwise, the SYSDBA account will not be secure, and unauthorized users can access databases on the server.

Adding a User

To add a new user, click on the Add User... button. This opens the User Configuration dialog box:

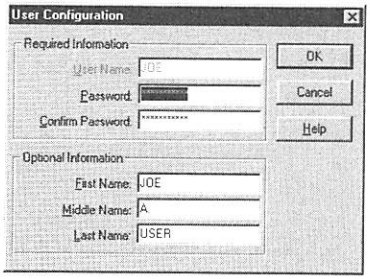


Type the new username in the username text field, and then type the user's password in both the Password and the Confirm Password text fields. Do not create usernames containing spaces. Add any desired optional information in the corresponding text fields, and then click on OK.

Note Usernames can be up to 31 characters long and are not case sensitive. Passwords *are* case sensitive. Only the first eight characters of the password are significant.

Modifying User Configurations

To change a user's password, select the username to highlight it, and then click on the Modify button. Alternatively, double-click on the username. This opens the User Configuration dialog box for that user. For example:



Change any of the entries except username and then click on OK. If you change the password, you must enter the same password in the Password text field and the Confirm Password text field.

You may not modify a username. The only way to change a username is to delete the user and then add a user with the new name.

Deleting a User

To remove a user's InterBase privileges, select the username and click the Delete button. A confirmation dialog box will inquire "Do you wish to delete user *username*?" If you choose OK, the user will no longer be authorized to access databases on the current server.

The SYSDBA may delete the SYSDBA user. In this case, no new users may be added to the database, and existing user configurations may not be modified. This is not recommended.

CHAPTER 8

Maintaining a Database

Database maintenance tasks include:

- Shutting down and restarting a database.
- Configuring database properties.
- Managing transaction recovery.
- Performing a database sweep.
- Validating and repairing a database.

Server Manager must be logged in to a server and connected to a database before performing any of these operations. All of these tasks are performed from the Database Maintenance Window except for validation, which is performed from the Server Manager window.

Selecting a Database

To perform database maintenance, first choose a database from the server/database tree:

1. Expand the server/database tree to display the database on which to perform maintenance.
2. In an expanded tree, click on the database name to highlight it. The highlighted database will be the one upon which Server Manager operates, referred to as the *current database*. When a database is highlighted, the server on which the database resides becomes the *current server*. Any actions of Server Manager then affect that server.

The name of the current database and its owner are displayed in the summary information area in the left pane, along with the versions of the InterBase server, client, and access method.

Shutting Down a Database

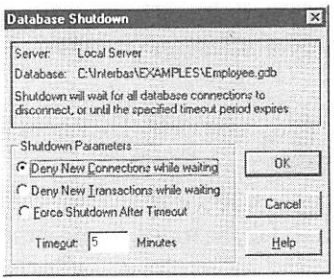
Maintaining a database often involves shutting it down. Only the SYSDBA or the owner of a database (the user who created it) may shut it down. The user who shut down the database then has *exclusive access* to the database.

Exclusive access to a database is required to:

- Validate and repair the database.
- Add a foreign key to a table in the database or drop a foreign key from a table in the database.
- Add a secondary database file.

After a database is shut down, the database owner and SYSDBA will still be able to connect to it, but any other user attempting to connect will get an error stating that the database is shut down.

To shut down the database, choose Maintenance | Database Shutdown from the Main Menu, toolbar, or pop-up menu. The Database Shutdown dialog box appears:



You can specify the following shutdown parameters:

- Timeout period in minutes. Server Manager will attempt to shut down the database during this period. Exactly how it does this depends on the other shutdown parameters.
- Deny new connections while waiting. This option allows all existing database connections to complete their operations unaffected. Server Manager will shut down the database after all processes disconnect from the database. At the end of the timeout period, if there are still active connections, then the database is *not* shut down.
- Deny new transactions while waiting. This option allows existing transactions to run to normal completion. Once transaction processing is com-

plete, Server Manager shuts down the database. Denying new transactions will also deny new database connections. At the end of the timeout period, if there are still active transactions, then the database is *not* shut down.

- Force shutdown. With this option, there are no restrictions on database transactions or connections. As soon as there are no processes connections to the database, Server Manager will shut down the database. At the end of the timeout period, if there are still active connections, Server Manager will roll back any uncommitted transactions, disconnect any users, and shut down the database.

Caution Forcing database shutdown will interfere with normal database operations, and should only be used after users have been given appropriate broadcast notification well in advance.

Denying New Connections

The Deny New Connections shutdown option prevents any new processes from connecting to the database during the timeout period. This enables current users to complete their work, while preventing others from beginning new work.

Suppose the SYSDBA needs to shut down database SALES.GDB at the end of the day (five hours from now) to perform routine maintenance. The Marketing department is currently using the database to generate important sales reports.

In this case, the SYSDBA would shut down SALES.GDB with the following parameters:

- Deny New Connections.
- Timeout of 300 minutes (five hours).

These parameters specify to deny any new database connections and to shut down the database any time during the next five hours when there are no more active connections.

Any users who are already connected to the database will be able to finish processing their sales reports, but new connections are denied. During the timeout period, the SYSDBA sends out periodic broadcast messages asking users to finish their work by 6 p.m.

When all users have disconnected, the database is shut down. If all users have not disconnected after five hours, then the database is not shut down. Because the shutdown is not critical, it is not forced.

It would be inappropriate to deny new transactions, since generating a report could require several transactions, and a user might be disconnected from the

database before completing all necessary transactions. It would also be inappropriate to force shutdown, since it might cause users to lose work.

Denying New Transactions

The Deny New Transactions option is the most restrictive shutdown option, since it will prevent any new transactions from starting against the database. This option will also prevent any new connections to the database.

Suppose the SYSDBA needs to perform critical operations that require shutdown of the database ORDERS.GDB. This is a database used by dozens of customer service representatives throughout the day to enter new orders and query existing orders.

At 5 p.m., the SYSDBA initiates a database shutdown of ORDERS.GDB with the following parameters:

- Deny New Transactions.
- Timeout of 60 minutes.

These parameters deny new transactions for the next hour. During that time, users can complete their current transactions before losing access to the database. Simply denying new connections would not be sufficient, since the shutdown can not afford to wait for users to disconnect from the database.

During this hour, the SYSDBA sends out periodic broadcast messages warning users that shutdown will happen at 6 p.m, and instructing them to complete their work. When all transactions have been completed, the database is shut down.

After an hour, if there are still any active transactions, Server Manager cancels the shutdown. Since the SYSDBA needs to perform database maintenance, and has sent out numerous warnings that a shutdown will occur, there is no choice but to force a shutdown.

Forcing Shutdown

If critical database maintenance requires a database to be shut down, and there are still active transactions, the SYSDBA can force shut down. This step should only be taken if broadcast messages have been sent out to users that shutdown will occur. If users have not heeded repeated warnings, then their work will be rolled back.

The Force Shutdown option does not deny new transactions or connections during the timeout period. If, at any time during the timeout period, there are no connections to the database, Server Manager will shut down the database.

At the end of the timeout period, if there are still active connections, Server Manager will roll back any uncommitted transactions, disconnect any processes, and shut down the database.

Restarting a Database

After a database is shut down, it must be restarted (brought back online) before users can access it.

To restart a database which has been shut down, choose Maintenance | Database Restart from the Main Menu, toolbar, or pop-up menu. The currently selected database will be brought back online immediately.

Viewing Active Database Connections

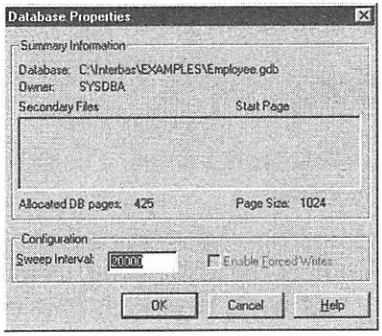
To view active database connections, choose Maintenance | Database Connections from the Main Menu, toolbar, or pop-up menu. The Active Database Connections dialog box appears.

The following dialog box will open:

The name of the current database and active users is shown.

Configuring Database Properties

To view and configure database properties, choose Maintenance | Database Properties.from the Main Menu, toolbar, or pop-up menu. The Database Properties dialog box will then appear:



This dialog box contains a Summary Information area that displays properties but does not allow modification of them and a Configuration area that does allow modification of the parameters.

The Summary Information area displays:

- Database name.
- User name of the database owner.
- Database Page Size.
- Number of allocated pages.
- Secondary file names and sizes.

The configuration area displays and allows modification of:

- Sweep interval.
- Enabling of forced writes.

Adjusting Database Sweeping

Sweeping a database is a systematic way of removing outdated records from the database. Periodic sweeping prevents a database from growing too large. However, sweeping can also slow system performance.

As a DBA, you can tune database sweeping, balancing its advantages and disadvantages to best satisfy users' needs.

Overview of Sweeping

InterBase uses a multi-generational architecture. This means that multiple versions of data records are stored directly on the data pages. When a record is updated or deleted, InterBase keeps a copy of the old state of the record and creates a new version. This can increase the size of a database.

To limit the growth of the database, InterBase performs *garbage collection*, which frees up space allocated to outdated versions of a record. Whenever a transaction accesses a record, outdated versions are garbage collected. Records that were rolled back are ignored by typical transactions and will not be garbage collected. To guarantee that all records are garbage collected, including those that were rolled back, InterBase periodically "sweeps" the database.

When sweeping a database, InterBase reads every record in the database. This forces garbage collection of outdated record versions as well as rolled back records. InterBase automatically sweeps a database at a set interval: by default, 20,000 transactions.

Because automatic sweeping is tied to a transaction, sweeping a database can affect application performance. For example, on a single-threaded platform, if a user's transaction coincides with the sweep interval, the user's process may stop while the sweep takes place. On multi-threaded platforms, the sweep can spawn a separate process so that it does not block other processes.

Tip Sweeping a database is not the only way to perform systematic garbage collection. Backing up a database achieves the same result because InterBase must read every record, an action that forces garbage collection throughout the database. As a result, regularly backing up and restoring a database can reduce the need to sweep. This enables you to maintain better application performance. For more information about the advantages of backing up and restoring, see Chapter 9: "Backing Up and Restoring a Database."

The Database Maintenance window enables you to control several aspects of database sweeping. You can:

- Change the automatic sweep interval.
- Disable automatic sweeping.
- Sweep a database immediately.

The first two functions are performed in the Database Properties dialog box. The last is performed with the Maintenance | Database Sweep menu command.

Setting the Sweep Interval

To set the automatic sweep interval, type the number of transactions between each database sweep in the Sweep Interval text field. For example, to set the sweep interval to 10,000 transactions, type 10000 in the text field.

Sweeping a database can affect transaction start-up if rolled back transactions exist in the database. As the time since the last sweep increases, the time for transaction start-up may also increase. Lowering the sweep interval may help reduce the time for transaction start-up.

On the other hand, frequent database sweeps may reduce application performance. Raising the sweep interval may help improve overall performance. The DBA should weigh the issues for the affected applications and decide whether the sweep interval provides the desired database performance.

Tip Unless the database contains many rolled back transactions, changing the sweep interval has little effect on database size. As a result, it is more common for a DBA to tune the database by disabling sweeping and performing it at specific times. These activities are described in the next two sections.

Disabling Automatic Sweeping

To disable automatic sweeping, set the sweep interval to 0. Disabling automatic sweeping is useful if:

- Maximum throughput is important. Transactions will never be delayed by sweeping.
- You want to schedule sweeping at specific times. Once automatic sweeping is disabled, you can manually sweep the database at a specified time.

Controlling Performance of Forced Writes

When an InterBase Server performs *forced writes* (also referred to as synchronous writes), it physically writes data to disk whenever the database performs an (internal) write operation.

If forced writes are not enabled, then even though InterBase performs a write, the data may not be physically written to disk, since operating systems buffer disk writes. If there is a system failure before the data is written to disk, then information can be lost.

Performing forced writes ensures data integrity and safety, but will slow performance. In particular, operations which involve data modification will be slower.

When forced writes are enabled an "X" appears in the box labeled "Enable Forced Writes" in the Database Properties dialog box. To disable forced writes, click on the check box to remove the "X".

InterBase for NetWare has a feature called the write-ahead log (WAL) which performs much the same function as forced writes. If a database is using WAL, then it is not necessary to enable forced writes to ensure data integrity. Therefore, forced writes will be automatically disabled when WAL is enabled for a database. Conversely, when WAL is disabled (dropped), forced writes are automatically re-enabled. You can manually override these defaults by enabling and disabling forced writes in the Database Properties dialog box.

Caution If neither WAL nor forced writes are enabled for a database, then the database will be subject to data loss if there is a hardware or other system failure. In general, it is best to have at least one of these features active.

Two-phase Commit and Transaction Recovery

When committing a transaction that spans multiple databases, InterBase automatically performs a two-phase commit. A *two-phase commit* guarantees that the transaction updates either all of the databases involved or none of them—data is never partially updated.

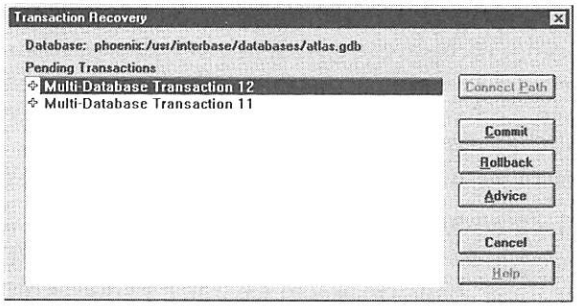
In the first phase of a two-phase commit, InterBase prepares each database for the commit by writing the changes from each *subtransaction* to the database. A subtransaction is the part of a multi-database transaction that involves only one database. In the second phase, InterBase marks each subtransaction as committed in the order that it was prepared.

If a two-phase commit fails during the second phase, some subtransactions will be committed and others will not be. A two-phase commit can fail if a network interruption or disk crash makes one or more databases unavailable. Failure of a two-phase commit causes *limbo transactions*, transactions that the server does not know whether to commit or roll back.

It is possible that a limbo transaction will make some records in a database inaccessible. To correct this, you must recover the transaction using Server Manager. *Recovering* a limbo transaction means committing it or rolling it back.

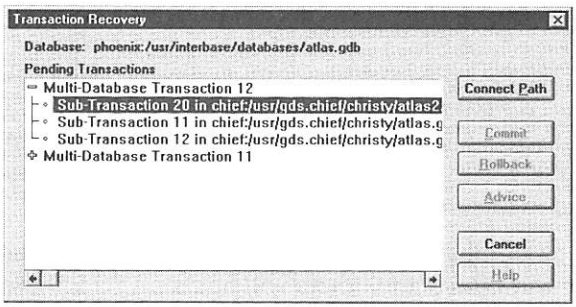
Recovering Transactions

To recover limbo transactions, choose Maintenance | Transaction Recovery from the Main Menu, toolbar, or pop-up menu. A dialog box will then display a list of limbo transactions that can then be operated upon to recover—that is, to commit or roll back:

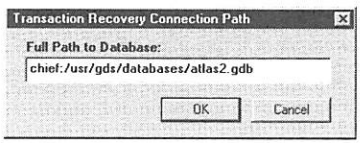


All the pending transactions in the database are listed in the scrolling area on the left side of the dialog box. Click on the "+" to display all the subtransactions of a transaction.

It is also possible to have a single database transaction that has been prepared and not committed. These transactions are displayed with a bullet to the left of the transaction. You can roll back or commit such transactions.



You can change the path of the database specified by each subtransaction by choosing Connect Path. The following dialog box will appear:



Enter the directory path of the other database involved in the subtransaction, including the server name and separator indicating communication protocol, then choose OK.

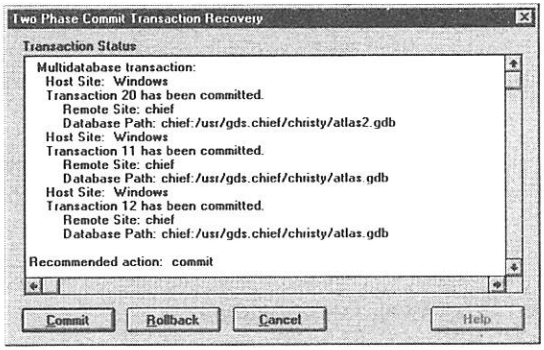
The information on the path to the database was stored when the client application attempted the commit. It is possible that the path and network protocol from that machine will not work from the client which is now running Server Manager. Before attempting to roll back or commit any transaction, confirm the path of all involved databases is correct.

To use TCP/IP, separate the server and directory path with a colon (:). To use Named Pipes/NetBEUI, precede the server name with either a double backslash (\\) or a double slash (//), and then separate the server name and directory path with either a backslash or a slash.

You can choose to either commit or roll back each transaction. To commit or roll back, select the desired transaction ID from the list and choose either Commit or Rollback.

Note Only entire transactions can be recovered, so the commit and rollback buttons will only be enabled when the main transaction is selected. They will be disabled when a subtransaction is selected.

You can also seek advice by choosing the Advice button. This dialog box will open:



This dialog box will display information on each subtransaction: whether it has been committed, the remote server name, and database name. At the bottom, an action will be recommended: either commit or roll back.

Server Manager analyzes the state of subtransactions by determining when the two-phase commit failed. If the first transactions are in limbo but later transactions are not, Server Manager assumes that the prepare phase did not complete. In this case, you are prompted to do a rollback.

Performing an Immediate Database Sweep

To perform a database sweep, choose Maintenance | Database Sweep from the Main Menu, toolbar, or pop-up menu.

This operation runs an immediate sweep of the database, releasing space held by records which were rolled back and by out-of-date record versions. Sweeps are also done automatically at a specified interval; see "Adjusting Database Sweeping," in this chapter.

Important Sweeping a database does not require it to be shut down. You can perform sweeping at any time, but it can impact system performance and should be done when it will least affect users.

Validating and Repairing a Database

In day-to-day operation, a database is sometimes subjected to events that pose minor problems to database structures. These events include:

- Abnormal termination of a database application. This does not affect the integrity of the database. When an application is canceled, committed data is preserved, and uncommitted changes are rolled back. If InterBase has already assigned a data page for the uncommitted changes, the page might be considered an *orphan page*. Orphan pages are unassigned disk space that should be returned to free space.
- Write errors in the operating system or hardware. These usually create a problem with database integrity. Write errors can result in "broken" or "lost" data structures, such as a database page or index. These corrupt data structures can make committed data unrecoverable.

You should validate a database:

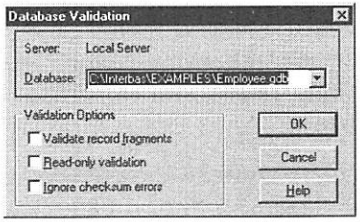
- Whenever a database backup is unsuccessful.
- Whenever an application receives a "corrupt database" error.
- Periodically, to monitor for corrupt data structures or misallocated space.
- Any time you suspect data corruption.

Note Database validation requires exclusive access to the database. Shut down a database to acquire exclusive access. If you do not have exclusive access to the database, you will get the error message

```
OBJECT database_name IS IN USE
```

For information on shutting down a database, see "Shutting Down a Database," in this chapter.

To validate a database, choose Maintenance | Database Validation from the Main Menu, toolbar, or pop-up menu. The following dialog box will open:



The name of the current database is displayed in the Database text field. Because there are some conditions such as a checksum error that will make it impossible to connect to a database, it is not necessary to connect to the database before performing a validation. If Server Manager is not connected to the database, you can enter the desired database name in the Database text field or select it from the drop-down list by clicking on the arrow to the right of the field.

When Server Manager validates a database it verifies the integrity of data structures. Specifically, it will:

- Report corrupt data structures.
- Report misallocated data pages.
- Return orphan pages to free space.

Validation Options

You can select three options with Database Validation:

- Validate record fragments
- Read-only validation
- Ignore checksum errors

By default, database validation reports and releases only page structures. When you select the Validate record fragments option, validation reports and releases record structures as well as page structures.

By default, validating a database updates it, if necessary. To prevent updating, select the Read-only validation option.

Handling Checksum Errors

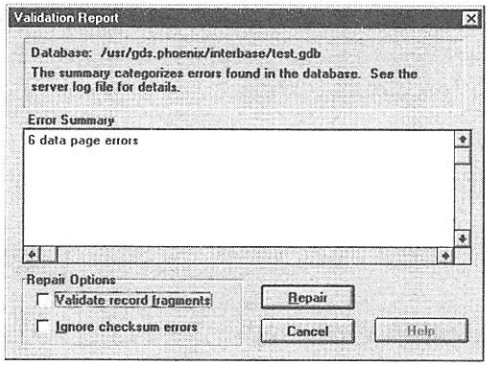
A checksum is a page-by-page analysis of data to verify its integrity. A bad checksum means that a database page has been randomly overwritten (for example, due to a system crash).

Checksum errors indicate data corruption. To repair a database that reports checksum errors, select the Ignore checksum errors option. This option enables Server Manager to ignore checksums when validating a database. Ignoring checksums allows successful validation of a corrupt database, but the affected data may be lost.

Caution Even if you can restore a mended database that reported checksum errors, the extent of data loss may be difficult to determine. If this is a concern, you may want to locate an earlier backup copy and restore the database from it.

Repairing a Corrupt Database

If a database contains errors, the following dialog box will open:



The errors encountered are summarized in the Error Summary area. The repair options you selected in the Database Validation dialog box will be selected in this dialog box also.

To repair the database, choose Repair. This will fix problems that cause records to be corrupt and mark corrupt structures. In subsequent operations (such as backing up), InterBase ignores the marked records.

Note Some corruptions are too serious for Server Manager to correct. These include corruptions to certain strategic structures, such as space allocation pages. In addition, Server Manager cannot fix certain checksum errors that are random by nature and not specifically associated with InterBase.

If you suspect you have a corrupt database, perform the following steps:

1. Make a copy of the database using an operating-system command. Do not use the InterBase Backup utility, because it cannot back up a database containing corrupt data.
2. Repair the copy database to mark corrupt structures. If Server Manager reports any checksum errors, validate and repair the database again, choosing the Ignore checksum errors option. It may be necessary to validate a database multiple times to correct all the errors.
3. Validate the database again, with the Read-only validation option selected. Note that free pages are no longer reported, and broken records are marked as damaged. Any records marked during repair are ignored when the database is backed up.
4. Back up the mended database with Server Manager. At this point, any damaged records are lost, since they were not included during the back up. For more information about database backup, see Chapter 9: "Backing Up and Restoring a Database."
5. Restore the database to rebuild indexes and other database structures. The restored database should now be free of corruption.
6. Verify that restoring the database fixed the problem by validating the restored database with the Read-only validation option.

CHAPTER 9

Backing Up and Restoring a Database

A database *backup* saves a database to a file on a hard disk or other storage medium. To protect a database from power failure, disk crashes, or other potential data loss, you should regularly back up the database. For additional safety, it is recommended to store the backup medium in a different physical location from the database server.

A database *restore* re-creates a database from a backup file.

Using the Backup and Restore Utilities

Operating systems usually include facilities to archive database files. Server Manager offers several advantages over such facilities, including:

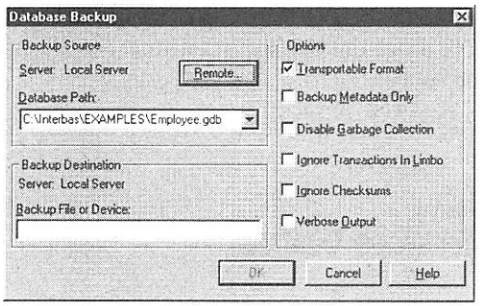
- Database performance can be improved. Backing up and restoring a database garbage-collects outdated records and balances indexes. The process also frees space occupied by deleted records and packs the remaining data, reducing database size. When you restore, you have the option of changing the database page size or distributing the database among multiple files or disks.
- Backups can run concurrently with other users. You need not shut down the database to run a back up. However, any data changes that occur after the back up begins are not recorded in the backup file. After you create a database backup, you can include it as part of a regular system backup.
- Multi-file databases are never partially backed up. If a database spans multiple files, Server Manager backs up either all the files or none.
- Data can be transferred to another operating system. Different computers have their own database file formats and therefore databases cannot simply be copied to a platform with a different operating system. If

desired, you can also make a backup in a generic format called a *transportable* backup that allows restoration to a server on a different operating system. Making transportable backups is highly recommended in heterogeneous environments.

Backing Up a Database

The database being backed up is referred to as the *source*. The file or device to which the database is being backed up is called the *destination* or *target*.

To back up a database, choose Tasks | Backup from the Main Menu, toolbar, or pop-up menu. The Database Backup dialog box appears:



This dialog box enables you to back up a database to a file or device. To perform a backup:

- Type the name of the source database (including path) in the Database Path text field in the Backup Source area. By default, the database to which Server Manager is currently connected is displayed. Other databases previously connected to can be displayed by clicking on the drop-down button to the right of the text field. To choose another database, select it from the list or type it in the text field.
- Type the name of the destination file or device in the text field in the lower left of the dialog box.
- Select the desired backup options, then choose OK to start the backup timer.

Server Manager will open a standard text display window to display status and any messages during the backup process.

Note Database files and backup files can have any name that is legal on the operating system; the. GDB and .GBK file extensions are InterBase conventions only.

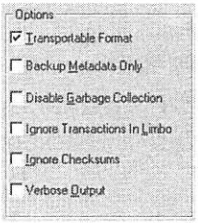
When creating a backup file, Server Manager stores the database as one file. You cannot split a large database among multiple backup files. A backup file will typically occupy less space than the database because it includes only the current version of data and incurs less overhead for data storage.

If you specify a backup file that already exists, Server Manager overwrites it. To avoid overwriting, specify a unique name for the backup file.

If a database spans multiple files, specify only the first file (the *primary* file) as the source. Server Manager uses the header page of each file to locate additional files, so the entire database can be backed up based on the primary file.

Backup Options

The backup options are indicated by check boxes on the right side of the Database Backup dialog box. If a check box has a ✔ inside, then the option is selected. If the box is empty, the option is not selected.



Transportable Format

To move a database to a machine with a different operating system from the machine on which the backup was performed, check the Transportable Format option. This option writes data in a generic format, enabling you to restore to any machine that supports InterBase.

To make a transportable backup:

1. Back up the database using transportable format by selecting the Transportable Format option in the Database Backup dialog box.
2. If you backed up to a removable medium, proceed to Step 3. If you created a backup file, use operating-system commands to copy the file to

tape, then load the contents of the tape onto another machine. Or copy it across a network to the other machine.

3. On the destination machine, restore the backup file. If restoring from a removable medium, such as tape, specify the device name instead of the backup file.

Back Up Metadata Only

When backing up a database, you can exclude its data, saving only its metadata. You might want to do this to:

- Retain a record of the metadata before it is modified.
- Create an empty copy of the database. The copy will have the same metadata but can be populated with different data.

To back up metadata only, select the Back Up Metadata Only option.

You can also extract a database's metadata using Windows ISQL. ISQL produces an SQL data definition (text) file containing SQL commands. Server Manager creates a backup file containing metadata only.

Disable Garbage Collection

By default, Server Manager performs garbage collection during backup. To prevent garbage collection during a backup, select the Disable Garbage Collection option.

Garbage collection physically erases old versions of records from disk. Generally, you will want Server Manager to perform garbage collection during backup.

You might not want to perform garbage collection during backup if there is data corruption in old record versions and you want to prevent InterBase from visiting those records during a backup.

Ignore Transactions in Limbo

To ignore limbo transactions during backup, select the Ignore Transactions in Limbo option.

When Server Manager ignores limbo transactions during backup, it ignores all record versions created by any limbo transaction, finds the most recently committed version of a record, and backs up that version.

Limbo transactions are usually caused by the failure of a two-phase commit. They can also exist due to system failure or when a single-database transaction is prepared.

Before backing up a database that contains limbo transactions, it is a good idea to perform transaction recovery, by choosing Maintenance | Transaction Recovery... in the Database Maintenance window.

Ignore Checksums

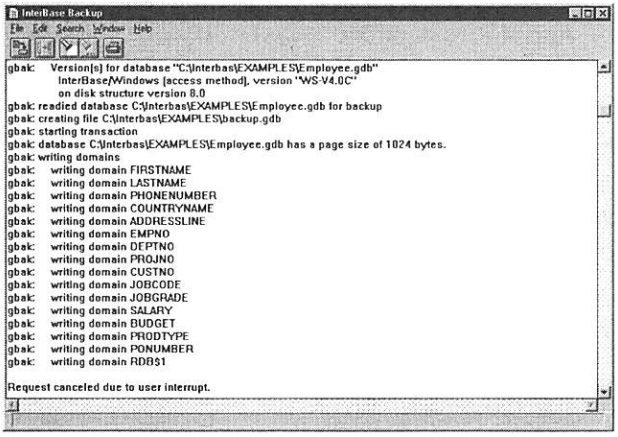
To ignore checksums during backup, select the Ignore Checksums option.

A checksum is a page-by-page analysis of data to verify its integrity. A bad checksum means that a data page has been randomly overwritten; for example, due to a system crash.

Checksum errors indicate data corruption, and InterBase normally prevents you from backing up a database if bad checksums are detected. Examine the data the next time you restore the database.

Verbose Output

To monitor the backup process as it runs, select the Verbose Output option. This option opens a standard text display window to display status messages on the screen. For example:

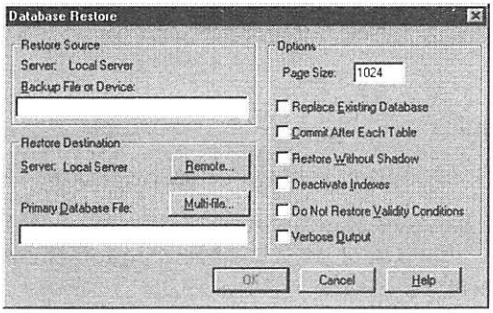

InterBase Backup
File Edit Search Window Help
gbak: Version[s] for database "C:\Interbas\EXAMPLES\Employee.gdb"
InterBase/Windows [access method], version "WS-V4.0C"
on disk structure version 8.0
gbak: readied database C:\Interbas\EXAMPLES\Employee.gdb for backup
gbak: creating file C:\Interbas\EXAMPLES\backup.gdb
gbak: starting transaction
gbak: database C:\Interbas\EXAMPLES\Employee.gdb has a page size of 1024 bytes.
gbak: writing domains
gbak: writing domain FIRSTNAME
gbak: writing domain LASTNAME
gbak: writing domain PHONENUMBER
gbak: writing domain COUNTRYNAME
gbak: writing domain ADDRESSLINE
gbak: writing domain EMPNO
gbak: writing domain DEPTNO
gbak: writing domain PROJNO
gbak: writing domain CUSTNO
gbak: writing domain JOBCODE
gbak: writing domain JOBGRADE
gbak: writing domain SALARY
gbak: writing domain BUDGET
gbak: writing domain PRODTYPE
gbak: writing domain PONUMBER
gbak: writing domain RDB$1
Request canceled due to user interrupt.

By default, the backup window displays the time that the backup process starts, the time it ends, and any error messages.

The standard text display window enables you to search for specific text, save the text to a file, and print the text.

Restoring a Database

To restore a database, choose Tasks | Restore from the bar menu or toolbar. The Database Restore dialog box will then appear:



This dialog box enables you to restore a database from a previously created backup file on the current server.

The backup file from which the database is being restored is called the *source*. The database being restored is called the *destination* or *target*.

To restore a database:

- Type the name of the source file or device on the current server in the Backup File or Device text field.

  To restore a database to more than one database file, click on the Multi-file button.... For more information about restoring to multiple database files, see "Restoring to Multiple Files," in this chapter.

- Type the name (including directory path) of the database to restore to in the Primary Database File text field.

- Type the page on which to start the restore in the Start Page field, and the page size, in bytes, in the Page Size text field. Typically, the starting page will be zero (0).

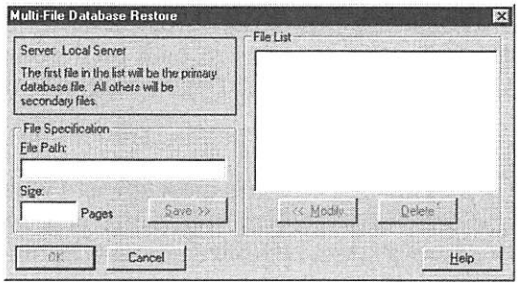
- Select the desired restore options, and choose OK to begin the restore.

Typically, a restored database occupies less disk space than it did before being backed up, but disk space requirements could change if the on-disk structure version changes. For information about the ODS, see "Upgrading to a New On-disk Structure," in this chapter.

Restoring to Multiple Files

You might want to restore a database to multiple files to distribute it among different disks, which provides more flexibility in allocating system resources.

To restore a database to multiple database files, click on the Multi-file button in the Database Restore dialog box. The following dialog box opens:



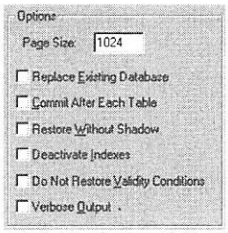
To specify the database files to restore to, type the file name in the File Path text field and then type the number of pages for that file in the text field below it. The minimum number of pages in a file is 200. Choose Save, and the file name will appear in the File List on the right side of the dialog box.

To modify one of the files in the list, select it and choose Modify. The selected file name will appear in the File Path text field, where you can edit it, and the associated number of pages will appear in the Pages text field. To delete a file, select it in the File List and choose the Delete button.

After entering all the names of the database files to restore to, choose OK to return to the Database Restore dialog box.

Restore Options

The restore options are shown in check boxes on the right side of the Database Restore dialog box. If a check box has an ✔ inside, then the option is selected. If the box is empty, the option is not selected.



Start Page

The Start Page is the page on which to start the restore. In most cases, this should be left as the default, zero.

Page Size

InterBase supports database page sizes of 1024, 2048, 4096, and 8192 bytes. The default is 1024 bytes. To change page size, back up the database and then restore it, modifying the Page Size field in the Database Restore dialog box.

Changing the page size can improve performance for the following reasons:

- Storing and retrieving BLOB data is most efficient when the entire BLOB fits on a single database page. If an application stores many BLOBs exceeding 1K, using a larger page size reduces the time for accessing BLOB data.
- InterBase performs better if rows do not span pages. If a database contains long rows of data, consider increasing the page size.
- If a database has a large index, increasing the database page size reduces the number of levels in the index hierarchy. Indexes work faster if their depth is kept to a minimum. Choose Tasks | Database Statistics to display index statistics, and consider increasing the page size if index depth is greater than two on any frequently used index.
- If most transactions involve only a few rows of data, a smaller page size may be appropriate, because less data needs to be passed back and forth and less memory is used by the disk cache.

Replace Existing Database

Server Manager will not overwrite an existing database file unless the Replace Existing Database option is selected. If you attempt to restore to an existing database name, and this option is not selected, the restore will fail.

Caution Replacing an existing database is discouraged. When restoring to an existing file name, a safer approach is to rename the existing database file, restore the database, then drop or archive the old database as needed.

Commit after Each Table

Normally, Server Manager restores all metadata before restoring any data. If you select the Commit After Each Table option, Server Manager restores the metadata and data for each table together, committing one table at a time.

This option is useful when you are having trouble restoring a backup file; for example, if the data is corrupt or invalid according to integrity constraints.

If you have a problem backup file, restoring the database one table at a time lets you recover some of the data intact. You can restore only the tables that precede the bad data; restoration fails the moment it encounters bad data.

Deactivate Indexes

Normally, InterBase rebuilds indexes when a database is restored. If the database contained duplicate values in a unique index when it was backed up, restoration will fail. Duplicate values can be introduced into a database if indexes were temporarily made inactive (for example, to allow insertion of many records or to rebalance an index).

To enable restoration to succeed in this case, select the Deactivate Indexes option. This makes indexes inactive and prevents them from rebuilding. Then eliminate the duplicate index values, and re-activate indexes through ALTER INDEX in Windows ISQL.

A unique index cannot be activated using the ALTER INDEX statement; a unique index must be dropped and then created again. For more information about activating indexes, see the *Language Reference*.

Note The Deactivate Indexes option is also useful for bringing a database online more quickly. Data access will be slower until indexes are rebuilt, but the database is available. After the database is restored, users can access it while indexes are reactivated.

Do Not Restore Validity Conditions

If you redefine validity constraints in a database where data is already entered, your data might no longer satisfy the validity constraints. You might not discover this until you try to restore the database, at which time an error message about invalid data appears.

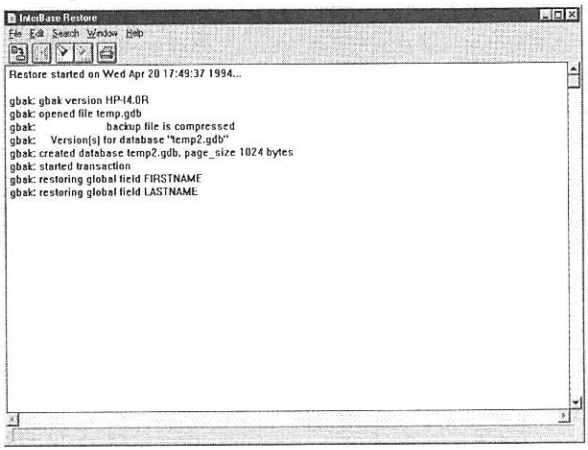
Caution Always make a copy of metadata before redefining it; for example, by extracting it using Windows ISQL.

To restore a database that contains invalid data, select the Do Not Restore Validity Conditions option. This option deletes validity constraints from the metadata. After the database is restored, change the data to make it valid according to the new integrity constraints. Then add back the constraints that were deleted.

This option is also useful if you plan to redefine the validity conditions after restoring the database. If you do so, thoroughly test the data after redefining any validity constraints.

Verbose Output

To monitor the restore process as it runs, select the Verbose Output option. This option will open a standard text display window to display status messages on the screen. For example:


InterBase Restore
File Edit Search Window Help
Restore started on Wed Apr 20 17:49:37 1994...
gbak: gbak version HP-I4.0R
gbak: opened file temp.gdb
gbak: backup file is compressed
gbak: Version(s) for database "temp2.gdb"
gbak: created database temp2.gdb, page_size 1024 bytes
gbak: started transaction
gbak: restoring global field FIRSTNAME
gbak: restoring global field LASTNAME

The standard text display window enables you to search for specific text, save the text to a file, and print the text. For an explanation of how to use the standard text display window, see Chapter 6: "Introduction to Server Manager."

Upgrading to a New On-disk Structure

New major releases of the InterBase server often contain changes to the on-disk structure (ODS). If the ODS has changed, and you want to take advantage of any new InterBase features, upgrade your databases to the new ODS.

You need not upgrade databases to use a new version of InterBase. The new versions can still access databases created with a previous version, but cannot take advantage of any new InterBase features.

To upgrade existing databases to a new ODS, perform the following steps:

1. Before installing the new version of InterBase, back up databases using the old version.
2. Install the new version of the InterBase server as described in *Installing and Running InterBase* for the platform.
3. Once the new version is installed, restore the databases with the new version of InterBase.

The restored databases will be able to use any new InterBase server features.

Index

A

accessing databases 83–87
adding
    comments in ISQL script files 57
    users 86
application development
    API applications 32
    Borland tools 31
    embedded SQL applications 31
    overview 31

B

backing up databases
    options 107
    preventing sweeping 108
    upgrading the on-disk structure 114–115
backup metadata only 108
BLOB data
    displaying in Windows ISQL 51
    improving access time 112
blob filters
    defined 30

C

case
    nomenclature 4
changing database page size 112
character sets in Windows ISQL 51
checksums
    ignoring 109
client/server development
    overview 19
column names
    nomenclature 4
columns
    defined 25
comments in ISQL scripts 57
commit after each table 113
configuring
    database properties 94–97
connecting to databases 84–85
    denying new connections 91
    in Windows ISQL 38
constraints
    naming 4
corrupt databases
    backing up by ignoring checksums 109
    repairing 102–103
creating
    databases in Windows ISQL 40–41
    ISQL script files 55
    multi-file databases 111

D

data types
    defined 25
database administration overview 15
database information displaying in Windows ISQL 52
database objects, naming 4
databases
    accessing 83–87
    backing up 106
    configuring properties 94–97
    connecting to 84–85
        in Windows ISQL 38
    creating
        in Windows ISQL 40–41
        multi-file 111
    deactivating indexes 113
    denying new connections 91–92
    denying new transactions 92
    disabling validity checking 113
    disconnecting from in Windows ISQL 42
    dropping in Windows ISQL 42
    extracting metadata 46–48, 108
    maintaining 89–103
    overview of building 24
    page size 112
    recovering transactions 97–99
    repairing 102–103
    replacing 112
    restarting 93
    restoring 110–114
    rolling back limbo transactions 99, 108
    shutting down 90–93

splitting into multiple files 111
starting page number 112
sweeping 94–96
immediately 100
upgrading the on-disk structure 114–115
validating 100–102
verifying structures 100–103
viewing active users 93
deactivating indexes 113
directories
path names 6
directories, path names 5
disable garbage collection 108
disabling validity checking 113
disconnecting from databases in Windows ISQL 42
domains
defined 27
dropping
users 87
dropping databases in Windows ISQL 42

E

event alerters
defined 29
executing ISQL script files 44
extracting metadata 46–48, 108

F

file names 4–6
files
.gbk extension 107
.gdb extension 107
creating multi-file databases 111
primary 111
secondary 111
filespec parameter 5

G

garbage collection, disabling 108
.gbk file extension 107
.gdb file extension 107

I

ignore checksums 109
ignore limbo transactions 108
indexes
correcting duplicate values 113
deactivating 113
defined 27
improving performance 112
information
where to find 17
integrity constraints
naming 4
InterBase
features 12
Local Server defined 9
multi-user server defined 9
shutting down on Windows 95 74
shutting down on Windows NT 76
starting on Windows 95 73
starting on Windows NT 74
ISQL
script files 55–69
adding comments in Windows ISQL 57
creating 55
executing 56
in Windows ISQL 44
SET statements 57–69

L

limbo transactions
ignoring 108
rolling back 99, 108
logging in to a server 83–84

M

maintaining databases 89–103
metadata
displaying in Windows ISQL 52–53
extracting 46–48, 108
multi-file databases 111
multi-file specifications 5

N

naming conventions 4–6
naming nodes 5
nodes
    naming 6
nodes, naming 5
nomenclature 4–6

O

on-disk structure 114–115

P

page size 112
parameters
    filespec 5
passwords, changing 87
path names 5, 6
platforms 4
primary file specifications 5
primary files 111
printing conventions (documentation) 2–4
properties
    viewing server 76

Q

queries 69

R

read-only database validation 101
record fragments, validating 101
recovering transactions 97–99
repairing databases 102–103
replacing databases 112
restarting databases 93
restore options 111–114
restoring databases 110–114
restrictions, nomenclature 4
rolling back limbo transactions 99, 108

S

secondary file specifications 5
secondary files 111
security 85–87
    adding users 86
    dropping users 87
Server Manager
    configuring database properties 94–97
    denying new connections 91–92
    denying new transactions 92
    maintaining databases 89–103
    menus 79
    pop-up menues 81
    restarting databases 93
    restoring databases 110–114
    security 85–87
    server/database tree 80
    shutting down databases 90–93
    starting 78
    summary information area 82
    Toolbar 79
    viewing active users 93
    window description 78
servers
    logging in 83–84
SET AUTODDL 58–59
SET BLOBDISPLAY 60–61
SET COUNT 61–62
SET ECHO 62–63
SET LIST 64
SET NAMES 65
SET PLAN 65–66
SET statements 57–69
SET STATS 66–67
SET TERM 68
SET TIME 69
shutting down databases 90–93
    denying new connections 91
    denying new transactions 92
SQL commands 14
SQL statements
    executing in Windows ISQL 42–45
starting
    InterBase on Windows 95 73
    InterBase on Windows NT 74
    options 73, 74
starting page number 112
statements
    example, printing conventions 3
stored procedures

defined 28
summary information area 82
sweeping databases 94–96
disabling 96
immediately 100
preventing during a backup 108
syntax
file name specifications 5
statements, printing conventions 3
SYSDBA 86

T

table names, nomenclature 4
tables
defined 25
extracting metadata 48
ToolBar 79
transactions
denying new transactions 92
limbo 99
recovering 97–99
rolling back limbo transactions 108
transportable format 107
triggers
defined 29
two-phase commit 97, 108

U

upgrading the on-disk structure 114–115
user names, changing 87
user-defined functions
defined 30
users
adding 86
dropping 87
viewing 93

V

validating
databases 100–102
record fragments 101
validity checking, disabling 113
verbose output 109, 114
views
defined 28
extracting metadata 48
naming 4

W

Windows ISQL
BLOB display 51
character sets 51
COMMIT 44, 56–57
connecting to databases 38
creating databases 40–41
disconnecting from databases 42
displaying
database information 52
metadata 52–53
dropping databases 42
exiting 39
extracting metadata 46–48
getting help 38
ROLLBACK 44, 56–57
saving
current sessions 46
SQL output 45
script files 55–69
adding comments 57
creating 55
executing 44, 56
settings
advanced 50–51
basic 49–50
changing 49–51
displaying 51–53
SQL statements
executing 42–45
starting 37
temporary files 39
using 37–53
version information 52
window description 38

Borland

Copyright © 1996 Borland International, Inc. All rights reserved. All Borland product names are trademarks of Borland International, Inc. Corporate Headquarters: 100 Borland Way, Scotts Valley, CA 95066-3249, (408) 431-1000. Internet: http://www.borland.com CompuServe: GO BORLAND. Offices in: Australia, Canada, France, Germany, Hong Kong, Japan, Latin America, Mexico, The Netherlands, Taiwan, and United Kingdom • Part # INT1340WW21774 • BOR 8790